2000

SICNN optimisation, two dimensional implementation and comparison

Grant Walker
Edith Cowan University

Edith Cowan University

Copyright Warning

SICNN Optimisation, Two dimensional Implementation and Comparison

By

Grant Walker

A Thesis Submitted in Partial Fulfilment of the Requirements for the Award of Bachelor of Engineering with Honours

EDITH COWAN UNIVERSITY

PERTH WESTERN AUSTRALIA

School of Engineering and Mathematics

Faculty of Communications, Health and Science

Edith Cowan University

November 2000

USE OF THESIS

The Use of Thesis statement is not included in this version of the thesis.

Abstract

The study investigates the process of optimisation, implementation and comparison of a Shunting Inhibitory Cellular Neural Network (SICNN) for Edge Detection.

Shunting inhibition is lateral inhibition where the inhibition function is nonlinear. Cellular Neural Networks are locally interconnected nonlinear, parallel networks which can exist as either discrete time or continuous networks. The name given to Cellular Neural Networks that use shunting inhibition as their nonlinear cell interactions are called Shunting Inhibitory Cellular Neural Networks.

This project report examines some existing edge detectors and thresholding techniques. Then it describes the optimisation of the connection weight matrix for SICNN with Complementary Output Processing and SICNN with Division Output Processing. The parameter values of this optimisation as well as the thresholding methods studied are used in software implementation of the SICNN.

This-two dimensional SICNN edge detector is then compared to some other common edge detectors, namely the Sobel and Canny detectors. It was found that the SICNN with complementary output processing performed as well or better than the two other detectors. The SICNN was also very flexible in being able to be easily modified to deal with different image conditions.

Acknowledgments

The first person I would like to acknowledge is Dr. Abdessalam Bouzerdoum for his guidance throughout the year, for his patience, his assistance in arranging facilities in which to work and, especially, his advice towards the end of the project about improving the final product. Our contact has always been pleasant and left me with hope where before there may have been despair.

Next I would like to thank my partner, Jane True. Through my 4 years of study she has been my support, my guide and my friend. She has also edited numerous essays turning them from mounds of drivel into what passed for documents, who says a degree in Fine Arts isn't useful! I thank her for her help and love.

My friends and family who have always had an interest in what I am doing, especially my mother whose glee at every one of my small successes has inspired me to work harder and perform better.

Most of all I'd like to dedicate this work to my grandfather, Alistair McCallum, who is my inspiration.

Declaration

I certify that this thesis does not, to the best of my knowledge and belief:

(i) *incorporate without acknowledgement any material previously submitted for a degree or diploma in any institution of higher learning;*

(ii) *contain any material previously published or written by another person except where due reference is made in the text; or*

(iii) *contain any defamatory material.*

Grant Walker

Abbreviations

β	Beta value for Kaiser window
f	SICNNs activation function
σ	standard deviation
$\alpha_{i\,j}$	Decay Factor
Δ_{xy}	gradient function in x/y direction
$\mathbf{a_b}$	area of black region
$\mathbf{a_w}$	area of white regions
C	Connection Weight Template
c	contrast of the signal/image
Cnew	DOP connection matrix
CNN	Cellular Neural Network
COP	Complementary Output Processing
dB	decibels
DOP	Division Ouput Processing
$\mathbf{E_{actual}}$	edge pixels in input image
$\mathbf{E_{dir}}$	Edge direction
$\mathbf{E_{found}}$	edge pixels found in SICNN output
$\mathbf{E_{mag}}$	Edge magnitude
ESNR	Edge Strength-to-Noise Ratio
FOM	Figure of Merit
HR	Hit Rate
$\mathbf{I_{ij}}$	input to thje ijth SICNN cell
$\mathbf{I_o}$	mean input/image intensity
max	maximum
PDvFA	Probability of Detection versus False Alarm

SAR	Synthetic Aperture Radar
SICNN	Shunting Inhibitory Cellular Neural Networks
SNR	Singal-to-Noise Ratio
v_b	grey levels of black region
v_w	grey levels of white region
w_{ij}	Connection Weight Matrix
x_{ij}	state of the ijth SICNN cell
Y	length of Template Matrix

Table of contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Vision systems

Within the eye light falls on the retina creating images. The retina is a sheet of neurons including a layer of photoreceptors, neurons specialised to measure light intensity and generate signals the rest of the nervous system can understand.

At this first processing step each photoreceptor generates a signal that is dependent on the intensity of light falling on it. Bright light causes the photoreceptor to generate a greater signal than do dark areas. These signals generated by the photoreceptors are processed in a number of ways by a variety of interactions among the neurons in the retina. One of those interactions is Lateral inhibition.

Figure 1.1 illustrates the signal processing that takes place in the retina.

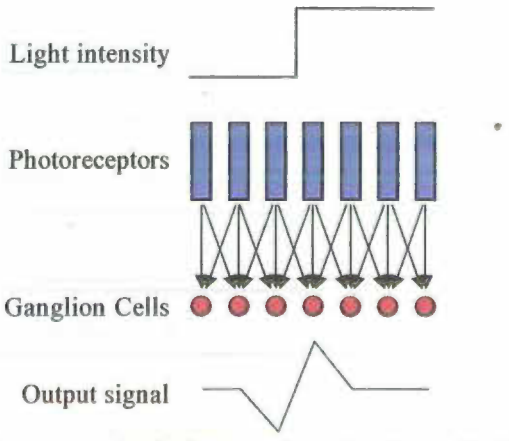


Figure 1.1 – Retinal processing using Lateral inhibition.

Photoreceptors generate a signal depending on the intensity of light falling on it. The circles are the ganglion cells, output neurons. The lines show the output from the photoreceptors going to the ganglion cells with the output of the photoreceptors adjacent also sending a signal which inhibits the excitation of the ganglion cell.

It is this laterally spread inhibition which gives lateral "inhibition networks" their name.

The signal of the output neurons is shown at the bottom of the diagram.

Just to the left of the edge the increased light intensity of the photoreceptor to its right side has an enhanced inhibitory effect. This causes a dip in the output signal.

Just to the right of the edge the increased main photoreceptor output with the smaller inhibitory effect of the photoreceptor to its left causes an increase in the level of signal output. Far to the left and far to the right of the edge the output neurons are excited by the overlying photoreceptors and inhibited by the adjacent photoreceptors. The network is organised so that equal illumination of exciting and inhibiting receptors equals out; the output neurons far from the edge will have the same, zero, signal.

This neural processing means the brain is not seeing exactly what light intensity is at each point on the retina but is instead sending information about which regions of the retina have edges, how large the edge is and whether the change in intensity is increasing or decreasing.

Lateral inhibition is the basis for many optical illusions and these can be used to demonstrate the phenomena. An example of this is the Hermann grid seen in the figure below.

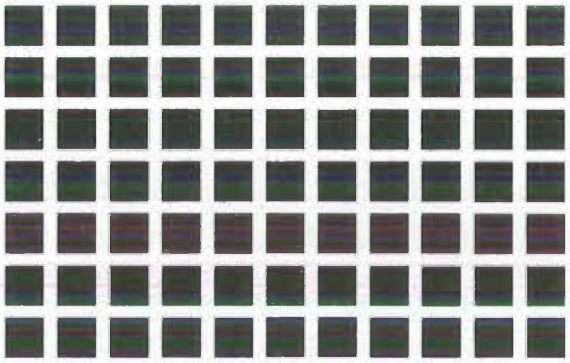
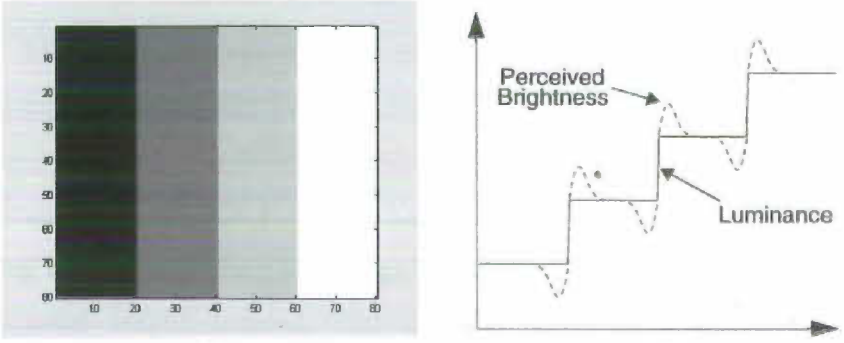


Figure 1.2 – The Hermann grid illusion

The intersections of the white lines in the Hermann grid appear darker than the region far away from the intersection. This is due to there being light coming from 4 sides at the intersection but only two sides in the regions between the intersections. The extra sides of light at the intersection inhibit the excitation of the intersection region causing it to appear darker.

Ernst Mach (Mach, 1886a; Mach 1886b) used lateral inhibition to explain the phenomena of Mach bands. Mach bands are light and dark bands adjacent to a luminance change. This illusory effect is demonstrated below.



(a) (b)

Figure 1.3 – (a) An image with bands of various intensity light and dark bands can be seen next adjacent to each transition (b) a plot showing actual luminance vs perceived brightness; From Pontecorvo (1998)

Dark bands appear on the dark side of the edge and light bands on the light side of the edge. The perceived brightness differs from the actual luminance due to inhibition.

However, the first extensive studies on inhibition were performed by Hartline and Ratliff on the compound eye of the horseshoe crab (Hartline & Ratliff, 1957; Hartline & Ratliff, 1958; Ratliff et al., 1963). The horseshoe crab is not actually a crab but is instead a water insect. The studies found that the neurons that were in the eyes of these insects matched the theoretical signal response proposed for lateral inhibition. The roles of lateral inhibition as outlined by Srinivasan et al. (1982) include;

- Redundancy removal to improve the efficiency on the supply of information through the optic nerve (Barlow, 1981)

- Removal of the DC bias in the input – to increase sensitivity (Brodie et al., 1978).

- De-blurring and edge enhancement e.g. the Mach bands phenomena.

- Predictive coding to improve efficiency and resolution (Srinivasan et al., 1982)

1.2 Cellular Neural Networks (CNN)

Cellular Neural Networks (CNNs) were introduced by Chua and Yang in 1988. They are nonlinear, parallel networks which can exist as either discrete time or continuous networks. The array of cells making up a CNN are locally interconnected and regularly repeated and they may be multi-layered.

The basic structure of a CNN is a cell. Cells in the array are designated C(i, j) with the i denoting the row and the j denoting the column. Each cell is connected to the r cells in its local neighbourhood. The neighbourhood of C(i,j) is called the r-neighbourhood.

Each cell looks the same and is a nonlinear dynamic subsystem which can be described by a nonlinear differential equation of the form.

$$\frac{dx_j}{dt} = g\left[x_j(t)\right] + \sum_{k \in N_j} A_{\mu_i}\left(x_j\big|_{(t-\tau,t)}, y_j\big|_{(t-\tau,t)}, p_j^A\right) \qquad \text{Eqn 1.1}$$

$$+ \sum_{k \in N_j} B_{v_i}\left(x_j\big|_{(t-\tau,t)}, I_j\big|_{(t-\tau,t)}, p_j^B\right) + u_j(t)$$

$$y_j(t) = f\left(x_j\big|_{(t-\tau,t)}\right) \qquad \text{Eqn 1.2}$$

The local interconnection of the cells gives the CNN its unique nature.

CNNs are ideally suited to being VLSI implementation due to the local nature of their interconnections. The CNNs regular architecture allows for large arrays to be designed easily. CNNs also work with analog interconnections allowing devices to be constructed using the advantages of analog computing – High speed, low power and small footprint.

There are many variants of CNNs defined. Generally variants involve changing the cell activation function, making the grid structure non-regular and varying the cell template matrices over time.

1.3 Shunting Inhibitory Cellular Neural Networks (SICNN)

To model some neural cells and visual phenomena linear lateral inhibition is insufficient. When the membrane conductance is controlled by the synaptic voltage of neighbouring cells, as it is in a typical neuron, the equation describing the lateral inhibitory neural network becomes non-linear. The equivalent electrical circuit for a neuron is shown in Figure 1.4 (Bouzerdoum & Pinter, 1993).

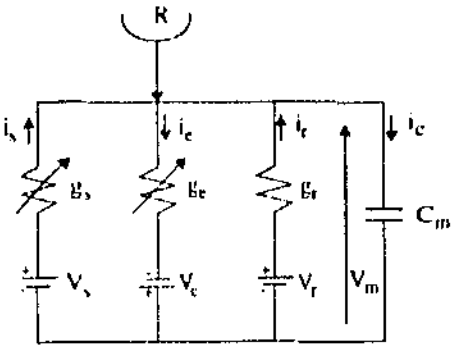


Figure 1.4 – Equivalent electrical model of a neuron; From Bouzerdoum and Pinter(1993)

Specific types of cellular neural networks have been developed that allow the use of non-linear lateral inhibition.

Many different types of non-linear processing elements are available within electronics. Amplifiers can provide multiplication or division and diodes can provide exponential functions. Cellular neural networks, which use multiplicative processing, are called Shunting Inhibitory Cellular Neural Networks (SICNNs).

They have been used by Pinter (1983a; 1983b) to explain selectivity for small objects of visual units in the ventral nerve of insects. They have also been used to explain adaptation of the receptive field spatial organisation and the spatial modulation transfer function.

Non-linear neural networks have also been used in image processing for image enhancement (Jernigan & Mcclean 1992; Paradis & Jerigan 1994) and for motion detection.

As artificial vision networks SICNNs are extremely effective as they are more resistant to multiplicative noise. Multiplicative noise has a standard deviation that is proportional to received intensity which means that peaks of the multiplicative noise look like larger edges than do additive noise peaks.

SICNN response is based on relative edge to intensity ratio. This allows the SICNN to have equivalent performance in high and low intensity areas under multiplicative noise.

Synthetic aperture radar (SAR) images and other coherent imaging processes (such as laser and sonar images) are plagued by a special type of noise called speckle noise. Research has shown that speckle noise is a multiplicative noise process inherent in the coherent imaging process. Due to the multiplicative noise performance of SICNNs they would be very effective as an edge detection process for these coherent images.

Shunting Inhibitory Cellular Neural Networks are biologically inspired networks with great promise as information processors for pattern recognition, vision and image processing tasks.

1.4 SICNN's and Edge detection

As previously explained lateral inhibition within the retinal neurons in a biological system exists to reduce the flow of information required to be processed at the brain.

Similar functions for lateral inhibition networks are practical for artificial vision systems.

Images of 640x480 pixels and 8-bit intensity quantisation consume over 2.45 Mbits. Multiple frames per second and colour images increase the required processing rates of vision systems greatly, requiring massively parallel processing arrays of high clock frequencies to be able to process the image data.

SICNNs ability to process data in time and space allows information on edges and motion to be output. Sending only this information rather than all the visual information reduces the data rate which in turn reduces processing requirements for artificial vision systems. Designs using SICNNs built with analog VLSI can provide real time processing.

Edge detection is the first process in many image processing tasks including segmentation. The output from a SICNN is ready for further complex processing operations.

The majority of edge detectors use linear models which provide adequate response to additive noise but not to multiplicative noise. They require complicated pre-filtering to lower the level of multiplicative noise which adds an extra step to image processing. The SICNNs ability to cope better under multiplicative noise conditions means they are ideal to use as the initial processing step in segmentation of coherent imaging systems.

In this thesis we investigate SICNNs as edge detectors. Edge detection is often the first and most important stage in many vision systems.

1.5 Project Outline

The objectives of this project are to:-

1. Implement and optimise three SICNN architectures
2. Develop and implement a two-dimensional SICNN architecture in Matlab
3. Compare performance of two-dimensional SICNN with conventional edge detectors.

The three architectures that will be implemented are the standard SICNN and the standard SICNN with complementary output processing, and SICNN with division output processing.

The standard SICNN has already been implemented and optimised to some extent in previous works conducted by Ward and by Pontecorvo (1998). The complementary output processing architecture was implemented but not optimised and the division output processing had not been implemented or optimised.

A two-dimensional SICNN architecture has previously been demonstrated but with limited capacity for someone not familiar with SICNNs to use. Thresholding for this

architecture is an important component and is covered in some detail within this thesis; see chapter 3 for details.

1.6 Thesis Organisation

This thesis is organised as follows.

Chapter 2 outlines many standard edge detector designs as well as explaining the theory behind the Sobel and Canny detectors used in the comparison with the SICNN edge detector. It also describes edge detection using SICNN edge detectors. It outlines the current designs and the previous work conducted on optimising the designs by Ward and Pontecorvo (1998).

Chapter 3 discusses the various techniques in post-processing, the edge processing enhancements and the important topic of thresholding. It explains a number of the global thresholding methods and some of the local thresholding methods.

Analysis of the evaluation of the edge detectors is explained in chapter 4. This chapter outlines the criteria which will be used to optimise the SICNN edge detectors and compare the Sobel and Canny edge detectors with the SICNN edge detectors in chapter 7.

Chapter 5 covers the method and the results of the optimisation of the SICNN with complementary output processing and the SICNN with division output processing.

Chapter 6 describes the software implementation of the SICNN as a Matlab toolbox using the information about optimised parameters from chapter 5.

Chapter 7 compares the SICNN with complementary output processing with the Sobel and the Canny edge detectors.

Chapter 8 presents the conclusions of the project and points the way forward to areas of further investigation.

Chapter 2
Edge Detection

Edge detection is one of the most common processes in image processing. It is common and useful because edges form the outline of an object and therefore can provide information about area, perimeter and shape. Computer vision is generally about identification and classification of objects and the information that edge detection provides is vital to this task.

Edge detection is part of a process called segmentation which is identification of regions within an image. After edge detection further processing can be used to determine what each region represents.

Edge detection is actually the process of determining the edge pixels, whereas edge enhancement increases the contrast between the edges and the background so the edges become more visible. Edge tracing is also used to follow the edges and collect them into a list.

When talking about the theory of edge detection most traditional methods talk about the ideal step edge. This is a change in grey level at exactly one point. The greater the step the easier it is to detect the edge. Without noise the step can be clearly determined.

Unfortunately in real world situations this ideal step is not realistic. Due to digitisation, the image is most often sampled with the edge across a number of pixels. Most edges that happen in nature are not exact. They can be considered to be a ramp with grey levels moving from one grey level to the other grey level. This may cause

the edge to appear over a number of pixels. Exact edges may be captured across a pixel causing three grey levels to represent one exact edge.

Another issue is noise. Many factors such as light intensity, motion, temperature, dust and lens effects can make two pixels that would normally be at exactly the same grey level to have different levels in the image. Noise means that ideal edges are never encountered in real images.

2.1 Derivative Based Edge detectors

Edges are characterised as areas of rapid change of grey level. Derivative operators are sensitive to this and can operate as an edge detector. The rate of change of grey levels is large near an edge and small in other areas.

As images are 2 dimensional level changes in both directions must be considered. For this reason partial derivatives of the image, with respect to the directions x and y, are used.

An estimate of edge direction can be determined by using the result of the partial derivatives of x and y as vectors and computing the vector sum. The operator used is the gradient operator which is a two dimensional vector.

$$\Delta I(x,y) = \left(\frac{\delta I}{\delta x}, \frac{\delta I}{\delta y}\right) \qquad \text{Eqn 2.1}$$

Due to the gradient function being a continuous function and the native data, being sampled, we must use something which approximates the derivative. This is the difference operation. To implement the difference operation we can take the difference between 2 adjacent pixels. For the horizontal direction;

$$\Delta_{x1} I(x,y) = I(x,y) - I(x-1,y) \qquad \text{Eqn 2.2}$$

For the vertical direction;

$$\Delta_{y1} I(x, y) = I(x, y) - I(x, y-1) \qquad \text{Eqn 2.3}$$

The problem with this approach is that it gives an approximation for the gradient at (x-1/2, y-1/2).

To get the result at (x, y) we could use for the horizontal direction;

$$\Delta_{x1} I(x, y) = I(x+1, y) - I(x-1, y) \qquad \text{Eqn 2.4}$$

For the vertical direction;

$$\Delta_{y2} I(x, y) = I(x, y+1) - I(x, y-1) \qquad \text{Eqn 2.5}$$

This operator gets a gradient at (x, y) but ignores the value of the pixel at (x, y).

To get a composite value for the pixel taking into account both horizontal and vertical result the magnitude of the edge response is calculated as follows.

$$E_{mag}(x, y) = \sqrt{\left(\frac{\delta I}{\delta x}\right)^2 + \left(\frac{\delta I}{\delta y}\right)^2} \qquad \text{Eqn 2.6}$$

There is also an edge direction that can be determined.

$$E_{dir}(x, y) = \arctan\left(\frac{\delta I/\delta y}{\delta I/\delta x}\right) \qquad \text{Eqn 2.7}$$

This approach of separate horizontal and vertical processing and combining is seen throughout edge detection and helps to reduce computational complexity and assists in determining edge direction.

2.2 Template Based Edge Detectors

Template based edge detection uses small templates as a model of an edge. The model can either be an attempt to model the level changes in the edge or an attempt to approximate a derivative operator, the latter appearing to be the most common.

There are many different template based edge detectors but the one that will be used for comparison here is the Sobel template.

2.2.1 Sobel

The Sobel edge detector uses convolution masks having the following form.

$$S_x = \begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ 1 & 0 & 1 \end{matrix} \qquad S_y = \begin{matrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{matrix}$$

These templates are really an approximation of the gradient of the pixel at the centre and are the equivalent of applying the Δ_1 gradient operator to each 2x2 portion of the 3x3 region and then averaging the result. The two components S_x and S_y correspond to the horizontal and vertical component of the edge.

Each S_x and S_y mask is convolved with the image I producing outputs I_x and I_y.

Magnitude of each pixel is calculated using the following formula.

$$E_{mag}(x,y)=\sqrt{I_x(x,y)^2+I_y(x,y)} \qquad \text{Eqn 2.7}$$

The direction of the edge can also be determined.

$$E_{dir}(x,y) = \arctan\left(I_x / I_y\right) \qquad \text{Eqn 2.8}$$

After determining the magnitude the edge map is then thresholded to give distinct black and white edges.

2.3 Other Edge Detectors

2.3.1 Canny

Canny's specification of the criteria for an optimum edge detector was that it possess:-

- Quality detection – find all edges
- Good localisation – the smallest possible distance between found edges and actual edges
- Single response to an edge

Canny defined these mathematically as a series of equations. The Canny edge detector creates a convolution filter that is based on the optimisation of these equations. The filter would smooth the noise and locate the edge.

To achieve the first criteria meant that the signal-to-noise ratio (SNR) and the localisation criteria needed to be maximised simultaneously.

$$SNR = \frac{\left| \int_{-W}^{W} g(-x)\, h(x)\, dx \right|}{\sigma_n \sqrt{\int_{-W}^{W} h^2(x)}} \qquad \text{Eqn 2.9}$$

Localisation is the inverse of the standard deviation of the spatial spread of detected edges about their true positions (Pontecorvo 98). The greater the localisation the better the edge detector performance.

$$Localisation = \frac{\left| \int_{-W}^{W} g'(-x) h'(x) dx \right|}{\sigma_n \sqrt{\int_{-W}^{W} h'^2(x) dx}} \qquad \text{Eqn 2.10}$$

The third criterion, the distance between noise generated peaks, is maximised. It has the form.

$$x_{max} = 2\pi \left(\frac{\int_{-\infty}^{\infty} h'(x) dx}{\int_{-\infty}^{\infty} h''(x) dx} \right)^{\frac{1}{2}} \qquad \text{Eqn 2.11}$$

Solving the criteria analytically is difficult but an efficient approximation is the first derivative of the Gaussian function.

$$G(x) = e^{\left(\frac{x^2}{2\sigma^2} \right)} \qquad \text{Eqn 2.12}$$

From Canny (86) it was found that the best filter form was the first derivative of the

Gaussian function.

$$G'(x)=\left(-\frac{x}{\sigma^2}\right)e^{\left(-\frac{x^2}{2\sigma^2}\right)} \qquad \text{Eqn 2.13}$$

In 2 dimensions a Gaussian is given by

$$G(x,y) = \sigma^2 e^{\left(-\frac{(x_2+y_2)}{2\sigma^2}\right)} \qquad \text{Eqn 2.14}$$

G(x,y) has derivatives in both the x and y directions. Taking these derivatives to make G'(x,y) and then convolving G' with image I will give an output image that has enhanced edges.

This two-dimensional convolution while easy to implement is expensive computationally. This two-dimensional convolution is equivalent to two one-dimensional Gaussian masks with the differentiation done separately using a differentiation convolution mask.

The process of the Canny edge detector is

1. Read image I

2. Create 1D Gaussian mask G with a parameter of the standard deviation

3. Create a 1D mask of the derivative of the Gaussian in the x and y directions G_x and G_y with the same standard deviation.

4. Convolve image with G along the rows to give I_x and along the columns to give I_y

5. Convolve I_x with G_x to give I_x' the x component of I convolved with the Gaussian

and convolve I_y with G_y to give I_y'

6. Compute the magnitude of the result at each pixel using the equation.

$$E_{mag}(x,y)=\sqrt{I_x'(x,y)^2+I_y'(x,y)^2} \qquad \text{Eqn 2.14}$$

This produces an edge image with large pixel values for edges small values for background. Simple thresholding techniques using global thresholds to show a pixel as white above threshold T and black below do not give very good results.

Canny thresholding uses thresholds based on the gradient of each pixel. Basically each of the edge pixels have a direction associated according to the formula below.

$$E_{dir}(x,y)=\arctan\left(\frac{I_x'(x,y)}{I_y'(x,y)}\right) \qquad \text{Eqn 2.14}$$

Edge pixels should have a gradient magnitude greater than the gradient magnitudes on either side of the edge. The final step with the Canny thesholding is called *nonmaximum suppression*, where pixels that are not local maxima are suppressed.

In the most common case, where the direction of the gradient of the pixel doesn't point in the horizontal or vertical directions, a linear interpolation of the gradients of the imaginary pixel adjacent and on the gridline is calculated.

The Figure 2.1 shows the case where the gradient of the central pixel does not point directly at an adjacent pixel.

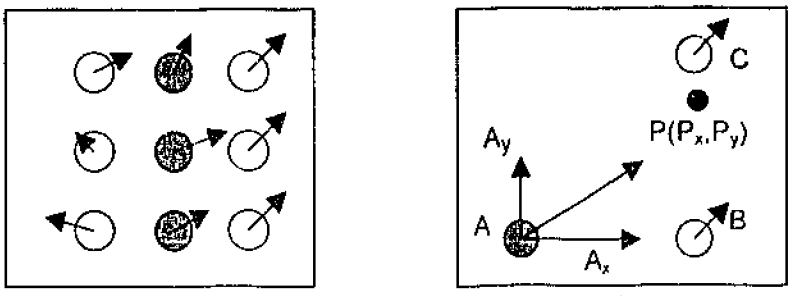


Figure 2.1. Nonmaximum suppression (a) Pixels with gradient directions not horizontal or vertical (b) Horizontal and Vertical vector components of gradient.

Pixel A has a gradient that points between pixel B and C.

The vector components of the gradient of A are A_x and A_y and B and C follow the same naming convention. The point P (P_x, P_y) to be calculated is the point at the intersection of a line drawn along the gradient direction of A and the gridline BC.

The gradient magnitude at point P(P_x, P_y) is estimated as

$$G = (P_y - C_y)Norm(C) + (B_y - P_y)Norm(B) \qquad \text{Eqn 2.15}$$

This is the case for vertical gridlines, the case for horizontal gridlines the Py value is replaced with the Px value and the C pixel is the pixel that is to the left of the B pixel.

$$G = (P_x - C_x)Norm(C) + (B_x - P_x)Norm(B) \qquad \text{Eqn 2.16}$$

This process is completed for every pixel in the canny output image and then the magnitude of the central pixel must be greater than both its neighbours'. If the magnitude is not greater, then its value is set to zero.

At this point the image still has grey levels. To remove these Canny suggests *Hysteresis thresholding*. Hysteresis thresholding is explained in Chapter 3.

2.4 Edge detection using SICNN

A feedforward shunting inhibitory cellular neural network is described by the following ordinary differential equation

$$\frac{dx_i}{dt} = I_i - \alpha_i x_i - \left[\sum w_j I_j\right] x_i \qquad \text{Eqn 2.17}$$

Where I_i is the input, α_i is the decay factor and w_j is the connection weight matrix. In steady state the output is given by

$$x_i = \frac{I_i}{\alpha_i + \sum w_j I_j} \qquad \text{Eqn 2.18}$$

This equation can be implemented using a convolution and a division operation.

The three architectures being compared are the standard SICNN, the SICNN with Complementary output processing (COP) and the SICNN with Division output processing (DOP)

The standard SICNN uses a single pass of a positive-X-Zero window. X-Zero windows are simply a way of describing the number of zeros in an asymmetric template matrix. Positive means that X zeros occur before M positive values and negative means that M positive values occur before X zeros.

All output processing is designed to increase the prominence of an edge peak from its standard prominence after a standard SICNN. There are a number of approaches to output processing using SICNNs. The two recognised as achieving the best response are Complementary Output Processing (COP) and Division Output Processing (DOP). These output processing methods are discussed in the output processing chapter 3.

2.5 Overview of previous work

This project work follows on from the Ph.D. thesis 'Edge detection and Enhancement using Shunting Inhibitory Cellular Neural Networks' by Carmine Pontecorvo.

This thesis studied SICNN edge detectors as compared with other standard edge detectors.

Pontecorvo's study introduces a number of post-processing techniques that significantly improve the quality of the output of SICNNs, most notably the complementary output processing technique that is used in this thesis.

It also derives mathematically the output of the SICNNs including the shape of the edge response. Readers interested in a thorough coverage of these aspects of SICNN are referred to Pontecorvo (1998).

Some areas that affect the SICNN performance were briefly examined. For example, the decay factor, α, that maximises the peak response to noise ratio (PNR) was found to be directly proportional to the mean intensity.

The symmetric and asymmetric weight distributions were examined and it was found that the asymmetric window yields a better performance.

The optimum performance from the distribution of connection weights was found to come when the sum of the connection weights is one. A basic study of the connection weights was completed. This determined that the greater the β value of the Kaiser distribution used to generate the connection weights matrix the thicker the edge response. This led to decrease performance of the edge detector.

A quantitative analysis of connection weights, decay factor and window size was part of the thesis report 'SICNN Optimisation for Edge Detection and Image Enhancement' by James Ward.

Ward's thesis used the HR (hit rate) and PdvFA(probability of detection Vs false

alarm) tests, both described in chapter 5 of this thesis, to evaluate an optimum β value for the SICNN edge detector. The optimum β value was found to be 1.4.

The symmetry of the connection weights was examined again and the conclusion was that the asymmetric window was optimum.

The number of zeros in the odd asymmetric window was examined with the best performance of the edge detector found to be when the matrix was odd length and the number of zeros, X, equals.

$$X = \frac{Y}{2} - 0.5 \qquad \text{Eqn 2.19}$$

Where Y is the connection weights matrix length.

The optimum length of connection weights was found to be 11.

Attempts at determining a way to remove the mean intensity (I_o) from the SICNN output was unsuccessful. The optimum decay rate from the one-dimensional SICNN tests was found to be 1.6 times the mean intensity I_o.

Chapter 3
Post Processing

There are a number of different options for post processing the SICNN output.

There is post processing to enhance the output SICNN levels and these include Complementary Output Processing (COP) and Division Output Processing (DOP). These attempt to increase the peak edge response of the SICNN edge detector.

Output processing is also conducted to convert the output image to a black and white binary image with 0 representing background pixels and 1 representing edge pixels. This output processing is called thresholding.

3.1 Complementary output processing

This technique was originally developed in the during trial and error period of project experimentation (Pontecorvo, 1998). Then it was called Negative Edge Noise Reduction (NENR) technique due to its ability to increase edge peaks and suppress noise.

Complementary output processing performs 2 SICNNs, one with positive X-zero window and the other with negative X-zero window, and then complements one with the other. This has the effect of reducing the peaks due to noise in the output and therefore increasing the peak edge response.

3.2 Division output processing

Division output processing is a scheme that was devised by Dr. Abdesselam Bouzerdoum as a post-processing scheme that could increase output peak edge response. It uses division as its operation which produces output which is centred about 1. Subtraction of one from this output moves the centre to being about 0.

Division output processing again performs 2 SICNNs as COP does and then divides the output of one by the output of the other. This can have the effect of increasing the peak edge response leading to increased chance of edge detection.

3.3 Thresholding

Thesholding or grey-level segementation is the conversion between a grey-level image and a bilevel (monochrome) image. This bilevel image should contain all of the essential information concerning the number, position and shape of objects in an image while containing a lot less other information. Reducing the complexity of the data simplifies many recognition and classification procedures.

There are a number of different methods of thresholding but all of them make use of some method to determine a range of grey levels that constitute an edge. The level of each individual pixel is then compared to the threshold and a determination is made on whether it is a black or white pixel.

Thresholding can be done on a global or a local level. Global level thresholds determines a threshold across the entire image whereas local thresholds generally determine thresholds for each individual pixel.

The more advanced thresholding algorithms use some form of recursion to repeatedly revise the threshold until the output to some error function is minimised.

3.4 Global Thresholds

Global thresholds use one threshold across an entire image. They tend to be simpler to implement and quicker to process the image data. The problems are in images with high contrast changes or Gaussian shading. In these images edges in low intensity areas tend to be missed and too many edges are picked up in high intensity areas.

For many images global thresholding gives a good processing/performance balance.

3.4.1 Histogram percentage

The Histogram percentage algorithm creates a histogram of the grey levels within a processed image. A percentage of pixels that are edges within the image must be selected.

The algorithm then uses this percentage to calculate the number of pixels that should be selected.

M=Total_number_of_pixels_in_image*Percentage Eqn 3.1

The threshold is determined by counting down through the grey levels on the histogram until the M^{th} pixel is reached. This grey level is used as the threshold with the pixels of grey levels equal to or above being an edge and the rest being background.

A modified version of this algorithm is used within the edge.m matlab code for thresholding the edges after processing. The modification allows for the detection of weak edges and strong edges and includes the weak edges in the final output where they meet with the strong edges.

The major flaw in this thresholding mechanism is the arbitrary selection of a percentage of edge pixels in an image. The resulting edge map may include pixels

that are not significant due to this.

3.4.2 Histogram Two Peaks

The two peaks algorithm came from the observation that there are generally two peaks in a histogram. The threshold is determined from the low point between the two peaks.

Finding the first peak is simple by just looking for the bin with the largest value. Most times the second largest peak will be the bin inside the largest so some method of valuing peaks that are away from the first peak would be good.

One method used commonly is to multiply the histogram values by the square of the distance from the initial peak. This gives a preference to peaks distant from the initial peak.

Thus if the largest peak is j the second largest peak k is

$$k = \max\left((k-j)h(k)\right) \qquad \text{Eqn 3.2}$$

Working down from the second peak using the values from the original histogram allows the evaluation of the low point between the peaks. This grey level is used to threshold.

3.4.3 Histogram Hysteresis

A high threshold is selected T_h and a low threshold is also selected T_l. Any pixels with a magnitude above T_h are automatically edges and any pixels whose magnitude is greater than T_l and which are also connected to the high threshold edge pixels are marked as edges also.

This threshold scheme has been implemented as an option within the SICNN toolbox.

It is the thresholding scheme used by edge.m.

Other methods are available which provide local thresholding according to gaussian distributions but these are computationally complex and it is arguable whether they give any better output.

Global thresholding is difficult for any image where contrast in the image is limited. In these cases local thresholding is effective.

3.5 Local Thresholds

Local thresholding techniques are generally considered to give better results than global ones. Local thresholds generally work by generating a threshold for each pixel based on some function of the pixels in the local neighbourhood.

Unfortunately the local thresholds that have been tried in this project have not given a significantly better output as they tend to react in a more extreme way to noise in areas of low intensity. This is not desired as this tends to create edges in the background rather than outlining objects, as is the intention of edge detection.

Added to this problem is that SICNN edge detectors respond to relative changes in intensity level meaning that small edges in low intesity areas react at a similar level to larger changes in high intensity areas. This means a jump from 1 to 2 grey level will output the same as a jump from 20 to 40 grey level.

3.5 1 Moving Average

The moving average thresholding technique uses a moving average to determine a threshold. Output with levels above twice the moving average can be included in the output image.

3.5.2 Relaxation

The relaxtaion method of thresholding works by recursively including pixels in either the set of white or black pixels. A comparison is then made which looks at the pixels adjacent to the current pixel and if the pixel is found to be completely surrounded by pixles of a different colour the pixel is moved from one set to the other set.

This continues recurisvely until none of the pixels change over one recursion, indicating they are all correctly identified.

This method tends to produce larger areas of full colour

Chapter 4
Evaluation of Edge Detector

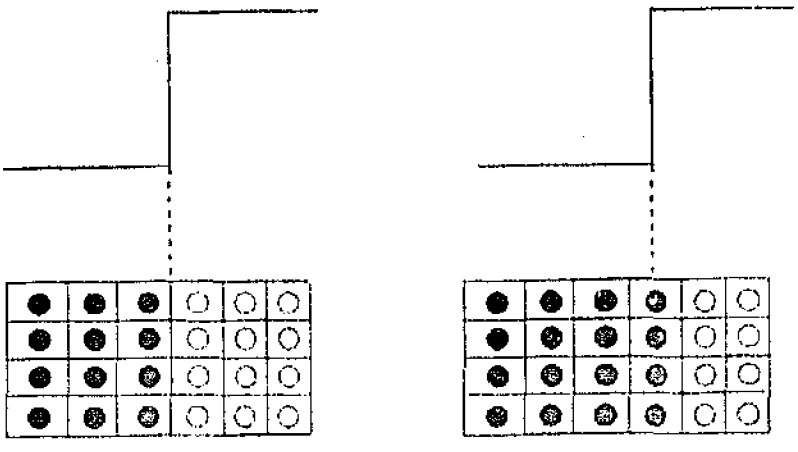
There are many different methods of evaluation of an edge detector. Currently the performance of the SICNN edge detector has been determined using the Hit-rate (HR) and the Probability of Detection versus False Alarm (PDvFA). These by their nature rate any errors as the same even though there may be localisation differences. They also don't take into account the thresholding that is required for any real edge detection algorithm. A more appropriate Figure of Merit (FOM) would include these issues

4.1 Edge models

Ideal edges have already been discussed but we have determined that these edges rarely occur due to:-

1. Objects not having a shape outline
2. Edges not occurring at the margins of a pixel
3. Noise

This means that to achieve realistic results for edge detection comparisons we must use realistic edges. Figure 5.1 demonstrates the problems with sampling.



(a) (b)

Figure 4.1 – The results of edge sampling (a) pixels align with edge (b) pixels do not align with edge; From (Parker ,1997, p5)

In part a) the image edge occurs right in the margins of a pixel causing an ideal edge. If we move the camera even a tiny bit to either side the edge falls midway in a pixel. This creates a pixel that has a grey level that is somewhere between the grey levels of the pixels on either side of the edge pixel. The actual grey level of the resultant pixel can be determined from the following equation.

$$v = \frac{(v_w a_w + v_b a_b)}{a_w + a_b} \qquad \text{Eqn 4.1}$$

Where v_w and v_b are the grey levels of the white and black levels, and a_w and a_b are the areas of the white and black parts of the edge pixel

In effect we have 2 "half" steps which correspond to exactly one edge within the image.

Noise is also an issue when it comes to determining the quality of an edge detector.

Noise cannot be predicted accurately because of its random nature and cannot even be measured accurately, so it is impossible to determine the contribution of the noise from the actual pixel data. Noise can, however, be characterised statistically by its effects on an image and has a mean and a standard deviation.

There are two types of noise specific to image analysis.

Signal independent noise is the noise that is added when an image is transmitted electronically from one spot to another. If A is the original image, N is the noise and B is the final image then

$$B=A+N \qquad \text{Eqn 4.2}$$

This is also termed additive noise.

A and N are unrelated to each other and although N could have any statistical properties it is assumed to be normally distributed about a mean of zero with a standard deviation σ.

The second type of noise for images is called signal dependent noise. This is noise where the level of the noise at each point in an image is a function of the grey level at that point. The grain seen in some photographs is an example of this type of noise.

$$B=A+f(A) \qquad \text{Eqn 4.3}$$

A specific type of signal dependent noise is where the noise has a standard deviation proportional to the grey level. This is called multiplicative noise.

All of these factors lead to a series of derived tests that can be used to determine the quality of an edge detection algorithm

4.2 Edge Strength-to-Noise Ratio

Noise is added to an ideal edge according to an Edge Strength Noise Ratio (ESNR) value. The ESNR equation is

$$ESNR = 20\log_{10}\left(\frac{2cI_o}{\sigma}\right) \qquad \text{Eqn 4.4}$$

Where

c = contrast value

I_o = mean intensity

σ = standard deviation of noise

Tests are run on ideal images with ESNR from 0-30dB.

At ESNR = 0, $\sigma=2cI_o$ or the size of the step edge, at ESNR=20 $\sigma=1/10*2cI_o$ one tenth of the size of the step edge. As the ESNR increases the edge becomes more defined. Figure 4.2 illustrates this.

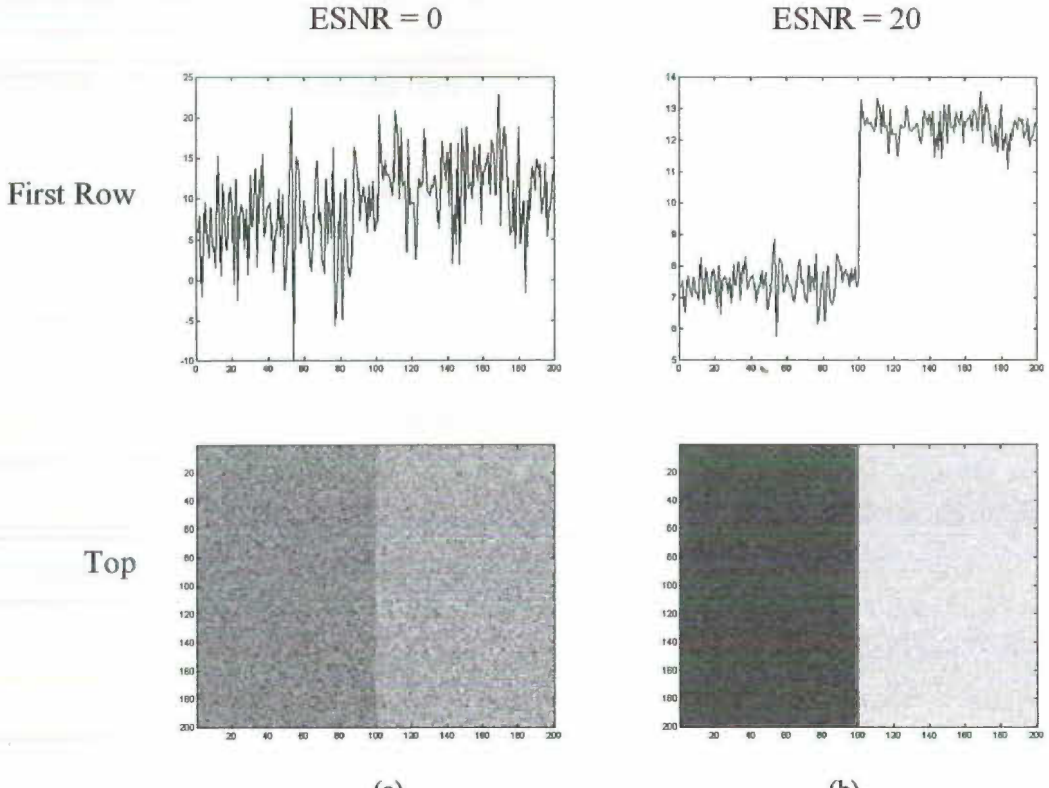


Figure 4.2 – Edges under (a) ESNR = 0 dB and (b) ESNR = 20 dB

4.3 Criteria

The evaluation tests use the SICNN edge detector to enhance the image and then they perform some form of thresholding to determine the edge pixels.

The general method of comparison of the performance of the various architectures being compared is via the following two measures.

- Hit Rate (HR) test
- Probability of Detection vs False Alarm (PDvFA)

4.3.1 Hit rate

As the test image only has one edge in each row the edge in the output is selected by finding the pixel with the maximum magnitude.

The hit rate test detects how often an edge detector correctly determines the pixel within an image that is an edge.

The position of each of the detected edges is compared to the position of the actual edge. If the positions match then a count is incremented.

After checking the detected edges in each row of the image the resulting count is normalised using the total rows in the image.

The result of this test is a percentage giving an indication of how often an edge would be detected in an image corrupted by a similar ESNR level.

The hit rate test is conducted over an increasing level of ESNR to show the rate of increase in hits as edge strength gets bigger.

4.3.2 Probability of Detection vs False Alarm

The Probability of Detection vs False Alarm (PDvFA) gives an understanding of the

level of edge peaks in an output SICNN processed image.

For a set ESNR the image is processed using SICNN. The output is progressively thresholded at rising threshold levels. At each thresholding level a count is done on the number of correctly detected edge pixels vs the number of falsely detected edge pixels.

The values for probability of detection are calculated as follows

$$PD = \frac{\sum\left(E_{found}(i,j) = E_{actual}(i,j)\right)}{\sum E_{actual}} \qquad \text{Eqn 4.5}$$

The values for false alarm are calculated as follows

$$FA = \frac{\sum\left(E_{found}(i,j) \neq E_{actual}(i,j)\right)}{\sum E_{found}} \qquad \text{Eqn 4.6}$$

The ideal values that would exist for the perfect edge detector would have a PD equalling 1 and FA equalling 0.

The PDvFA gives an indication of the level of elevation of an edge peak from the noise floor. High peaks at edges enable the edge detectors to operate under higher levels of noise.

This test is also a measure of the noise suppression of an edge detector. When the noise is suppressed the number of peaks from the noise floor decreases and the floor becomes less jagged.

Neither of these criteria measures two other important factors in evaluation of edge detectors, localisation to actual edges of incorrectly detected edge pixels and resolution of output image.

Localisation is how close a detected edge is from the actual edge. False edges can be detected anywhere on an image and a localisation figure of merit attempts to rate higher the edge detectors which, when they do incorrectly detect an edge, it is in the local area of an actual edge.

4.4.3 Pratt

One possibility is the Figure of Merit (FOM) defined by Pratt (1978). The aims of this figure of merit was to penalise the detector according to the square of the distance from the detected edge pixel to the actual edge pixel. The Pratt FOM has the following form:

$$FOM = \frac{\sum_{i=1}^{I_I} 1/1 + \alpha d(i)^2}{\max(I_A, I_I)} \qquad \text{Eqn 4.7}$$

I_A is the number of edge pixels found by the edge detector.

I_I is the number of edge pixels in the image.

d(i) is the distance between the i^{th} pixel of the actual edge and the one found by the edge detector.

α is used for scaling and is kept constant for a set of trials.

There are recognised problems with evaluation based on the Pratt FOM, such as lack of local edge coherence and no penalties for clustering of false alarms or missed edges. For optimisation using simple edge images, however, it is adequate. The Pratt FOM is implemented in the optimisation of COP and DOP chapter and is used to compare with the output of the HR test.

Chapter 5
Optimisation of COP and DOP SICNNs

This chapter outlines the methods that will be used to optimise the parameters for the complementary output processing and division output processing methods.

We have decided to use a slightly modified process based on the optimising procedures outlined by in previous studies (Ward, 1999; Pontecorvo 1998).

5.1 Method

The methods that will be used to optimise the parameters for SICNNs with complementary output processing and division output processing will be similar to the tests performed in the previous studies conducted on optimisation (Ward, 1999; Pontecorvo 1998).

The parameter to be optimised is the Kaiserβ value for the connection weight matrix C. A summary study of the optimum value of the decay factor will also be conducted. A decision was made to use the β based Kaiser window selection of optimum connection weights. Although this limits the allowable weights in the connection matrix and is likely to not determine the actual "best" weight matrix it simplifies the process. It will allow a quick selection of the optimum out of a range of β values which allows the project to continue on to comparison of the merits of SICNN edge detectors as against other standard edge detectors. It is recommended though that a more appropriate method of finding the optimum connection weights be based on

some genetic algorithm. This method would avoid bias in selection of particular weighting functions that will be assumed to have the best set of weights.

Unfortunately this method is complex requiring work on determining the evaluation function, the fitness function and how the genetic information about connection weights can be passed between generations of weight indexes. This study is outside the scope of this project as it would entail a full study of genetic algorithms and time requirements do not allow for this.

The method used by the previous studies for optimisation of connection weights was to initially use the HR test then confirm the results using the PDvFA test.

The Hit Rate test used a visual approach to narrow the region of β values likely to give the best edge output. The PDvFA test then confirmed this region.

More extensive tests were then undertaken using the HR test to narrow the region of β values again so that the value was determined to be between two integer values.

A final test that calculated the total area under the HR curve was then run to arrive at an optimum value.

The problem with this testing is that it doesn't take into account localisation of the determined edge peaks to the actual edges. The HR test only determines the maximum value and this also doesn't take into account the suitability of the schemes for thresholding.

To account for the localisation effect of the peaks another test will also be undertaken to give a peak value based on a localisation figure of merit, the Pratt figure of merit.

A comparison of the results of these two methods of connection weight optimisation will determine the relative merits of the HR test and complete the optimisation of the parameters.

5.2 Results

5.2.1 Complementary Output Processing Results

The COP Hit-rate test was run with β values ranging from 0-25 at intervals of one. Figure 5.1 shows the hit rate test results for 0, 5, 10, 15, 20 and 25 β values under noise ranging from 0-20 dB ESNR.

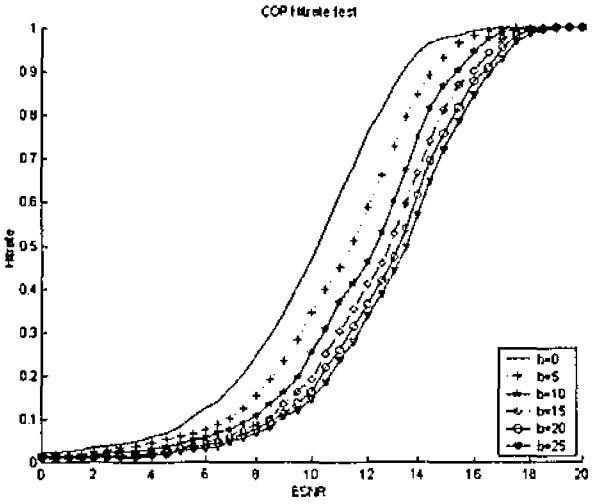


Figure 5.1 – Hit rate test for COP SICNN β =0, 5, 10, 15, 20, 25

Clearly the tendency as the β value increased the resulted in worse HR test figures. A plot of the sum of the HR results from 0-20 dB across β from 0-25 is shown in Figure 5.2. This graph demonstrates this even more dramatically.

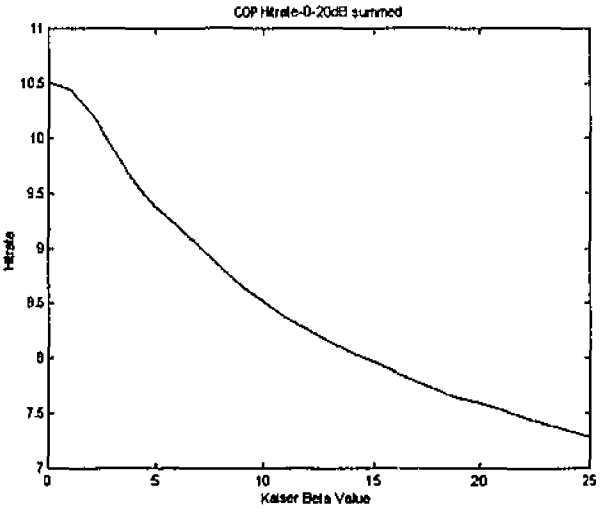


Figure 5.2 –Area under HR curve for COP SICNN β value 0-25

The optimum β value for this COP SICNN would be somewhere in the range 0-2.

A second HR test was done using β values from 0-2 at 0.1 intervals. A plot of the sums of the HR from 0-20dB for β from 0-2 appears in Figure 5.3.

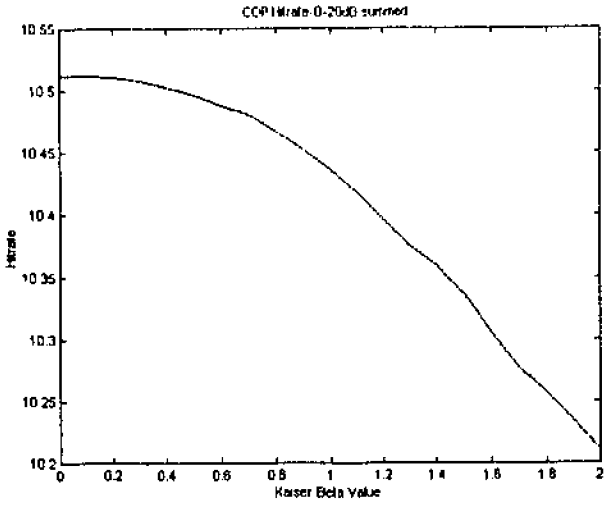
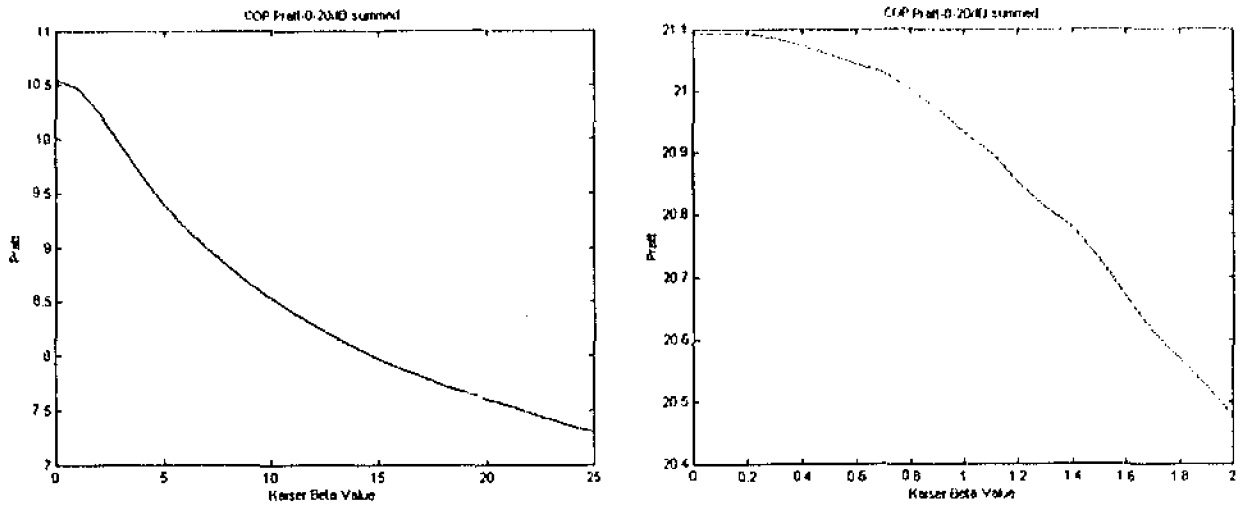


Figure 5.3 – Area under HR curve for COP SICNN β value 0-2

This test shows the improvement in the output of the HR test as the β value approaches zero. This seems to indicate an optimum βvalue of 0.

The conclusion to be drawn from this is that the optimum β value for the connection weights is β = 0. This conclusion is different from the result of the previous thesis which concluded that the optimum β value was 1.4.

Next confirmation of this result using the Pratt FOM was conducted. Figure 5.4 (a) shows the result of the ESNR 0-20 dB summed Pratt test over the range 0-25 β using the interval 1. Figure 5.4 (b) shows the result of the ESNR 0-20 dB summed Pratt test over the range 0-2 β using the interval 0.1



(a) (b)

Figure 5.4 – COP (a) Summed Pratt test 0-25 β (b) Summed Pratt test 0-2 β

This Pratt test confirms the results of the HR test in that the optimised β value for the COP SICNN is 0. It should be noted that the Pratt test is usually employed to test edge detectors with a threshold. The threshold use for this test was the same one used for the HR test which simply selects the maximum value along each processed row. Including a thresholding method in the test may achieve different results.

There are a number of differences between the two thesis runs that could account for the difference in the previous studies results and this current study.

First the sample size used in the previous thesis was 200x40, while the results for this thesis come from an image matrix of 1000x200. Having a greater sample size allows more accurate statistical information to be determined enabling more certainty in output results.

Second, for low ESNR values the SICNN output at the edge is almost indistinguishable from the output of the SICNN in the background. This means that the greater the size of the background the higher the chance that one of the background pixels will be selected as the edge pixel rather than the actual edge.

This is reflected in the change in the HR results at low noise between the previous

thesis results and this thesis.

Third, in James Wards thesis the HR for the COP SICNN edge detectors at 0dB ESNR was at 0.1 (See Figure 5.3, James Wards thesis). The output at 0dB ESNR of the COP SICNN in this thesis was less than 0.025. This is due to the increased sample size.

5.2.2 Division output processing

A mathematical simplification of the output of the Division Output Processing method can be achieved and this can be used to reduce the computational complexity.

5.2.2.1 Simplification of DOP

The output from the DOP method is

$$E_{out} = \frac{E_L}{E_R} - 1$$

$$E_{out} = \frac{E_L - E_R}{E_R}$$

The individual SICNN outputs are

$$E_L = \frac{I}{\alpha + C_L * I}$$

$$E_R = \frac{I}{\alpha + C_R * I}$$

$$E_{out} = \frac{I(\alpha + C_R * I) - I(\alpha + C_L * I)}{(\alpha + C_L * I)(\alpha + C_R * I)} \frac{(\alpha + C_R * I)}{I}$$

$$E_{out} = \frac{(C_R - C_L) * I}{\alpha + C_L * I}$$

$$E_{out} = \frac{C_{new} * I}{\alpha + C_L * I}$$

This effectively means that using DOP we remove the stability problems inherent in

using a weight index that has positive and negative components.

The new connection weights index can be modified to optimise DOP with the C_i acting as a contrast inhibition matrix.

A reasonable view would be that the Cnew matrix be designed so that in areas of similar intensity across all pixels Cnew evaluates to zero.

An effective Cnew that follows these design criteria would be.

$$[-1 \quad -1 \quad -1 \quad -1 \quad 1 \quad 1 \quad 1 \quad 1]$$

5.2.2.2 Optimisation of DOP

The DOP SICNN HR test was conducted by modifying the Cnew connection weights matrix. The β value was used to construct a set of Kaiser weights and then the first half of the matrix was converted to negative. The matrix length used was 10.

The contrast inhibition matrix was normalised to one and the weights were all the same. This made the contrast inhibition matrix 10 0.1 values.

Figure 5.5 shows the DOP HR test for β of 0, 5, 10, 15, 20 and 25 over ESNR in a range from 0-20dB.

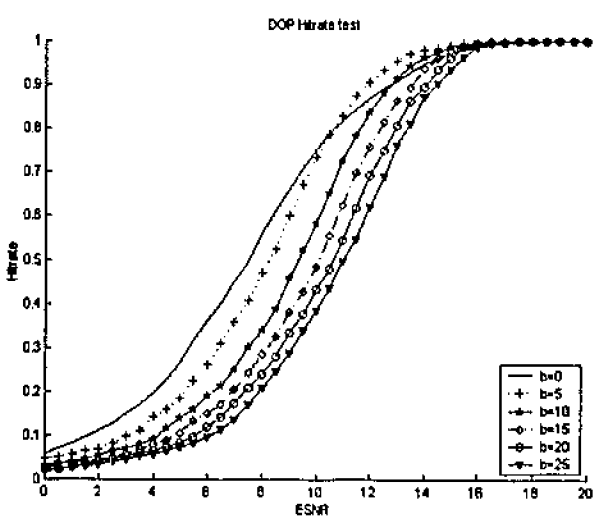


Figure 5.5 – Hit rate test for DOP SICNN β =0, 5, 10, 15, 20, 25

It is obvious in this graph that the performance of the detector declines as the β value increases above 10. This indicates the optimum connection weights βvalue is below 10.

Looking at Figure 5.6, which shows the summed HR area from 0-20dB, the optimum β value is in the range of 0 to 3.

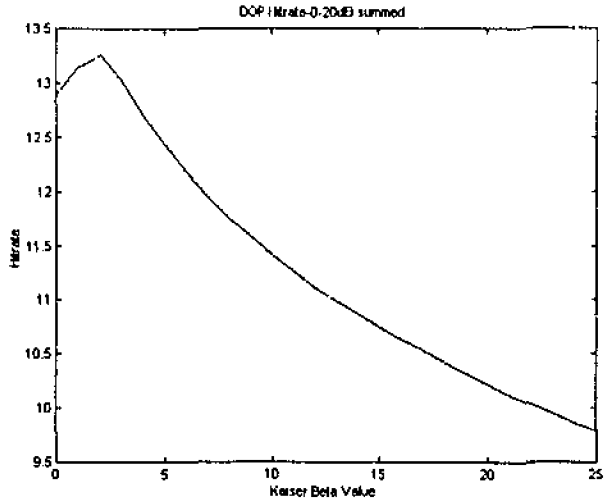


Figure 5.6 –Area under HR curve for DOP SICNN β value 0-25

Figure 5.7 shows the same summed area test run on intervals of 0.1 over the range 1 to 3. The optimum β value of the DOP SICNN from this graph is 1.7.

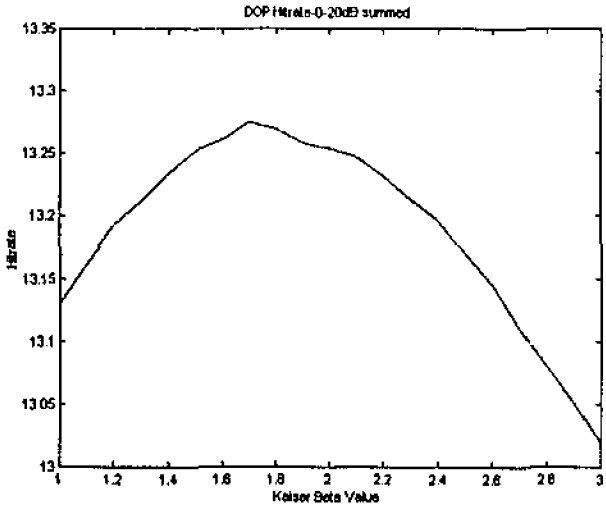
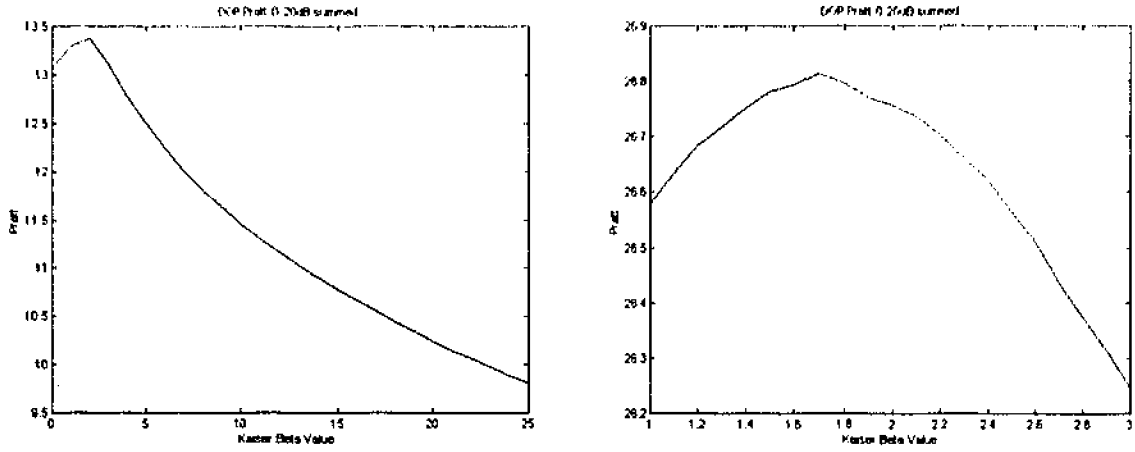


Figure 5.7 – Area under HR curve for DOP SICNN β value 0-2

Confirmation of this result using the Pratt FOM tests was then conducted. Figure 5.8 (a) shows the result of the ESNR 0-20 dB summed Pratt test over the range 0-25 β using the interval 1, Figure 5.8 (b)shows the result of the ESNR 0-20 dB summed Pratt test over the range 1-3 β using the interval 0.1



(a) (b)

Figure 5.8 – DOP (a) Summed Pratt test 0-25 β (b) Summed Pratt test 0-2 β

The Pratt FOM tests result in exactly the same optimum β value of 1.7 confirming the result from the HR test.

Due to the simplification of the Division Output processing SICNN it now requires only 2 convolutions and a division operation. This is considerably less processing than the complementary output processing SICNN which requires 2 convolutions, 2 divisions and a matrix subtraction to give similar levels of edge detection.

A plot of the results of the HR test for the optimum DOP and optimum COP SICNN appears in Figure 5.9.

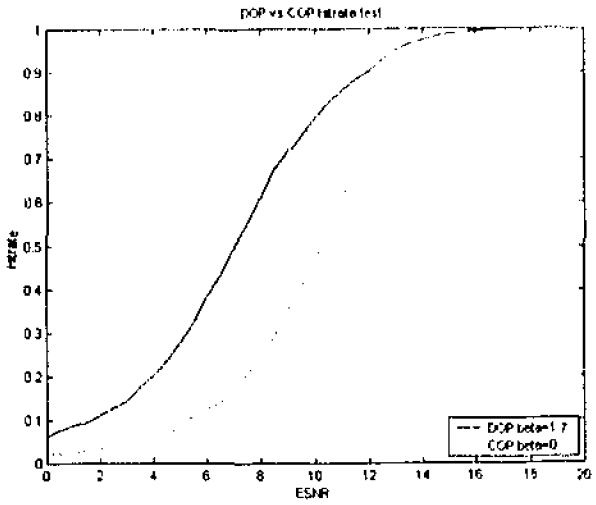


Figure 5.9 - HR Comparison of optimum COP and DOP SICNNs

The figure shows that the DOP SICNN significantly outperforms the COP SICNN having much higher performance in the ESNR region from 4 to 12.

This increased simplicity along with a better performance of the DOP SICNN means that it is the best method currently available for output processing SICNN edge detection.

5.2.3 Decay Factor

Analysis of tests in the previous thesis conducted on the output of the SICNN as mean intensity was varied showed peaks that were dependent on mean intensity I_0. Either side of this peak a sharp decline in output hit rate occurred but as the decay factor got higher the Hit Rate output stabilised. The graph from the thesis is in Figure 5.10.

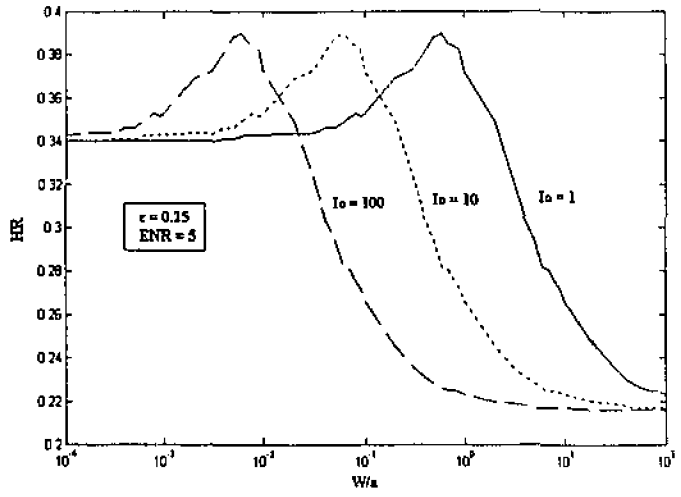


Figure 5.10 – SICNN output performance as Decay factor α is varied

A decay rate can be chosen which will enable the SICNN to operate in this region. This is not the optimum decay rate but instead gives good performance over a large range of mean intensities and contrasts conditions which are likely to exist in real world images.

In order to be operating in this region of operation the decay rate should be approximately twice the mean intensity over the entire image.

Tests conducted on real images where the decay factor was varied based upon the average intensity of the region about each pixel throughout an image created edge peaks in background areas that were not part of any objects. This is another reason to have a set decay factor value for an entire image.

5.3 Conclusion

This chapter analysed the HR results for complementary output processing and the results showed that the highest HR occurred when the connection weights matrix was constructed using a Kaiser β value of 0.

An optimised Kaiser β value for the creation of the connection weights matrix for the division output processing SICNN was also investigated and the result was a β value of 1.7.

Interestingly the Pratt test used to confirm the optimum values had exactly the same result as both of the HR tests. This confirms the value of the HR test in optimising the parameters of the connection weights.

The decay rate was selected based on the previous studies of the HR output of a SICNN in different mean intensity and contrast conditions and attempts to always operate the SICNN edge detector in the not optimum but more stable region of the

graph. This resulted in a decay rate of 2 times the mean intensity over the entire image.

The results of this optimisation will be implemented in the software implementation of the SICNN toolbox, which is presented in the next chapter, and the optimised values will improve the overall performance of the 2 dimensional SICNN edge detector.

Chapter 6
Software Implementation of SICNN

The initial aim of this project was to produce a single SICNN command that could be used for further investigation of the performance of two-dimensional SICNNs.

To do this it is important to allow for a highly customisable command that will allow the user to alter:-

1. The connection weights
2. The decay factor
3. Output processing style
4. Thresholding style.

6.1 Command I/O structure

The intention is to provide a command that can be used as a function and can also be used by itself in much the same way as the edge command within the image processing toolbox.

The connection weights may be supplied as one or two-dimensional matrices. The connection weights should not include any negative weights as the stability of the SICNN is not assured. If connection weights are not supplied a connection weight matrix, which has been optimised for the particular type of output processing, will be created and used.

The form of the output from the SICNN2d function will be dependent on the structure that it is being passed to. The SICNN2d can pass the output image, or the output image and the connection weight matrix, or in addition it can output the decay factor used.

The edge command is structured so that if an output matrix is not defined then a new figure will be created which will display the thresholded edge image.

6.2 Options

The decay factor can be selected and should be a positive value to ensure SICNN stability. Should the decay factor not be supplied then one is selected based on the optimum decay factor studies.

The output processing style will specify '' for no output processing, 'COP' for complementary output processing or 'DOP' for division output processing. Should the output processing style be omitted then the default selection will be no output processing.

Thresholding will automatically be performed using the following options:-

- '2p' indicates the two peaks histogram method,
- 'none' indicates no thresholding,
- 'edge' indicates use of the hysteresis histogram thresholding standard used by the edge.m command in the image toolbox..
- 'MA' indicates moving average thresholding.

6.3 Standard Matlab Toolbox structure

Creating a toolbox in Matlab allows execution of the toolboxes commands from within any working directory.

All of the .m files that are part of a toolbox are contained within the same directory. This directory can be anywhere although convention has it generally placed below the toolbox directory which is a directory immediately beneath the Matlab installation directory.

The directory that has been created must be included in the MATLABPATH variable. This variable is initially created from a list of directories contained in the toolbox/local/pathdef.m file. To add toolbox directories the line "'/MATLABR11/toolbox/toolbox_dir_name:',..."must be added to the end of the *Path Defined Here* section. The MATLABR11 directory name should be replaced by the Matlab installation directory name.

The next time Matlab loads the new path will be in the MATLABPATH variable.

This variable is used for two purposes; as a search path and a help file creation path.

When a function is typed at the Matlab command line the Matlab program follows the following process.

1. Checks for variable name in memory

2. Checks for function in memory

3. Checks for Matlab built in function

4. Checks current directory for filename.

5. Checks for toolbox function through the MATLABPATH

Any program in a local directory has precedence over MATLABPATH directory functions.

The path for MATLABPATH is also used in compiling the help window information.

Each directory in the MATLABPATH that contains .m functions should have a contents.m file.

The contents.m file has general information about the files contained within the directory. The initial Help screen that is displayed on the help window looks at the contents.m file in each directory in the MATLABPATH, extracts the first comment line, and compiles a path with "directory name – first line of contents.m file".

Double-clicking on the directory line displays the entire contents.m file.

The contents.m file can contain references to the individual .m files. To reference a .m file the name of the file should be at the beginning of the line, after the % comment symbol, and should be followed by a horizontal dash (-). After the dash a brief description can be written.

When these individual .m file lines are double-clicked the Matlab help system finds the file with the same name and checks that the name is repeated within the name.m file and as the first word, in capitals, of the first comment line of the .m file.

The entire comment section below this is then displayed as the help text.

It is useful when creating Matlab .m files to have knowledge of how Matlab processes the files. Script files are processed with Matlab loading a line at a time each time the script is run. Functions load the entire .m file into memory and then run the program from there. This makes functions execute considerably faster than script files. Bearing this in mind it is good practice to create functions within toolboxes as this speeds processing noticeably for more complex tasks.

6.4 Conclusion

The result of the software implementation is the toolbox that is outlined in Appendix C. This toolbox includes implementations of different thresholding methods which can be used to test the effectiveness of each of the thresholding styles.

The toolbox approach provides a self contained set of functions that can be used by people with only basic understanding of Matlab to explore the SICNN edge detectors and perhaps compare them with other types of thresholding styles or edge detector types.

Chapter 7
Comparison of 2D SICNN with Other Edge detectors

7.1 Methods of comparison

Evaluation of the output of edge detection algorithms with real images is difficult. Edge detectors can operate on images which have widely varying contrasts and background noise. Each edge detector has parameters which can be altered to optimise the edge detector for particular conditions.

The Canny detector allows the selection of different σ (standard deviation) values. It also allows for different high and low hysteresis thresholds.

The Sobel template edge detector can use larger matrices for greater noise suppression.

The SICNN edge detector can use different connection weights, connection weight lengths and decay factors to alter the performance and resolution.

For comparison the Canny and Sobel edge detectors will be used as implemented in Matlab. This means that to process images the edge.m Matlab function will only be passed the image and the detector type requiring the edge.m file to determine threshold levels.

The SICNN will use the COP method of output processing and use a positive 6-Zero

window of length 11 as the connection weight matrix. The β value used to create the window was 0. The reason the COP SICNN was used is that this component of the research was completed before the optimised performance of the simplified DOP SICNN was discovered.

The edge detectors will be evaluated using two methods.

The first method is objective and based on the Pratt figure of merit mentioned previously in chapter 4.

Image 1 will be applied to each of the edge detectors and then the output of the edge detector will be compared with the actual edge map to give a Pratt FOM. The edge detectors will be ranked according to this.

The second method will be subjective. A series of images will be passed through each detector. The output will then be shown to ten people who will be asked to rank the images according to the following question:

"Which edge image most accurately represents the objects in the actual image?"

This question has been chosen to avoid the selection process being just arbitrary, as it would be if the question "Which edge image is better?" was asked. As the process of edge detection is mostly used as the first step in segmentation and then object identification it makes sense to evaluate the edge detector performance with this goal in mind.

7.2 Objective comparison results

The objective test used the following image:

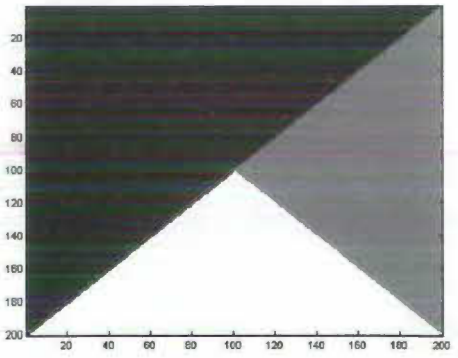


Figure 7.1 – Test image used in Pratt FOM testing – image 1

This image was designed to have a couple of different step sizes. The black region is at intensity 1, the grey region is at intensity 3 and the white area is at intensity 7. The edges are at angles to allow the 2D edge detectors to be tested.

Different levels of noise, with ESNR based on an edge size of 2, were then added to the image and then the image was processed by the Sobel, Canny and COP SICNN edge detectors. Hysteresis Histogram Thresholding was set at 0.95 of the image pixels as background for the Canny and SICNN edge detectors. The output images are presented in Appendix A.

The output Pratt FOM values under different levels of noise are in Table 7.1

ESNR	Sobel	Canny	COP SICNN
0	0.275	0.154	0.028
5	0.459	0.290	0.276
10	0.800	0.881	0.721
15	0.949	0.961	0.955
20	0.948	0.957	0.956

Table 7.1 – Pratt Figure of Merit results

The SICNN fares well at higher ESNR levels performing better than the Sobel edge detector and very close to the Canny detector. At low ESNR the SICNN is by far the worst, detecting edges poorly.

The thresholding method affected the results for both the Canny and the SICNN COP edge detector. The hysteresis histogram threshold, which requires a certain number of edge pixels to be selected for the output edge image, combined with the Pratt FOM penalising incorrect edges based on the square of the distance to the actual edge, mean that the pixels that were found in the noise greatly reduced the Pratt results.

At high ESNR the Canny edge detector seems to reduce in Pratt FOM output. This could be due to the post-processing morphological thinning operations built into the Canny edge.m implementation. Overall both the SICNN and Canny edge detector seem to peak at around the same value of 0.96 whereas the Sobel peaks at around 0.95.

This demonstrates the importance of having an appropriate thresholding scheme when operating under noisy conditions.

7.3 Subjective Comparison

The subjective test was conducted using the 11 images in appendix A. The processed images and the raw results of the survey also appear in appendix A.

Points were given to each of the images, 3 for being selected as best, 2 for being selected as second best and 1 for being selected last. Some statistics on the results are shown in Table 7.2.

Statistic	Sobel	Canny	COP SICNN
Total	179	232	249
Average	1.627	2.109	2.264
Selected Best	18	43	49
Selected Worst	59	31	20

Table 7.2 – Summary statistics for subjective assessment.

The test shows a clear distinction between the COP SICNN and the Canny and Sobel edge detectors.

The Sobel edge detector rated worst in every statistic, being selected as the worst detector in over half of the 110 possible selections. The output from the Sobel edge detector tended to miss important edges that could be used to determine an object's identity. The areas where the Sobel edge detector did well involved images with high contrast very few objects and with low levels of background intensity changes as demonstrated in image 2, Appendix B. The larger numbers of pixels required in the output images of the SICNN and Canny detectors meant that the actual edges in the image were thick or there was superfluous background information that obscured the object.

Figure 7.2 shows the number of times each edge detector was selected as best for each image.

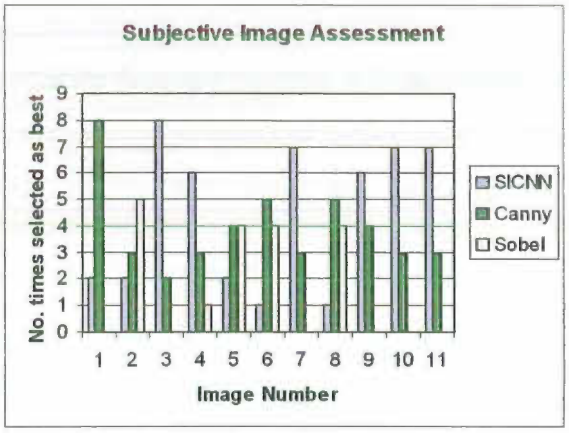


Figure 7.2 – Subjective image assessment: Images selected as best

For 5 of the 11 images each edge detector was selected as best by at least one person. This shows the volatility of subjective assessment.

In total the SICNN was selected most times as best in images 3, 4, 7, 9, 10 and 11. Canny was selected as best for 1, 5, 6, 8 and Sobel was selected best for 2 and 5.

The average score for the image results are in Table 7.3.

Image Number	1	2	3	4	5	6	7	8	9	10	11
SICNN	2.1	2.1	2.8	2.5	1.9	1.4	2.7	1.6	2.4	2.7	2.7
Canny	2.8	1.7	2	1.7	2.1	2.3	1.9	2.2	2.1	2.2	2.2
Sobel	1.1	2.2	1.2	1.8	2	2.3	1.4	2.2	1.5	1.1	1.1

Table 7.3 – Average subjective score for each image.

A major problem with the SICNN edge detector mentioned by the subjects was the thick lines around objects. The subjects mentioned that the thick lines made the objects less clearly identifiable. The thick lines were not present in the Canny edge detector as the non maximum suppression and the morphological thinning stage which is included in the implementation ensure that edge lines are only one pixel wide.

Further tests showed that by reducing the length of the connection weights matrix the width of lines was immediately reduced. Figure 7.3 shows the Lenna image that has been processed by the SICNN edge detector under three different length weights matrixes. The larger the matrix the thicker the lines but the greater the noise suppression.

(a) (b) (c)

Figure 7.3 – The image Lenna processed by SICNN with connection weight matrix length (a) 3 (b) 7 (c) 11.

In addition the hysteresis threshold was reduced to select only 5% of image pixels as edge pixels and this further improved the clarity of the edges.

It seems that both of these parameters can be used to significantly modify the output of the SICNN edge detector and should the detector be implemented as a plug-in for Photoshop these are the particular parameters that should be modifiable.

The modification of the threshold limits for the hysteresis thresholds is necessary to reduce the edges detected in some images and increase the number of edge pixels in other images with many objects.

Alternatively one of the other thresholding methods, including the effective local thresholds available in the SICNN toolbox, may give better image output.

7.4 Conclusion

Overall the comparison of the SICNN edge detector against Canny and Sobel edge detectors has found the SICNN edge detector performance is comparable.

The Canny edge detector shone in the Pratt FOM with the SICNN matching the output of the Canny at high ESNR.

When it came to the subjective assessment the SICNN edge detector performed exceptionally, being better than the Canny or the Sobel detection. The ability to modify the threshold parameters and connection weight length provided the flexibility for the SICNN edge detector to perform in a variety of conditions.

Clearly when the performance of the SICNN edge detector is married with its small computational cost it becomes very attractive as the first stage in the segmentation process.

The ability to implement the SICNN edge detector in analog VLSI fairly simply ensures this edge detection method has a future in artificial vision systems.

Chapter 8
Conclusions

8.1 Summary of Results

This project has examined the issues in edge detection and specifically the SICNN edge detector and its performance in comparison to other edge detectors.

Studies of edge detectors confirmed that most of the best performing and most advanced edge detectors involve many stages and significant levels of processing overhead. Simple solutions, such as template operators do not perform particularly well on most images.

Thresholding methods were analysed and this is an area that still needs more research. SICNN edge detectors produce images which have edge peaks which are significantly above the background level. Standard histogram style thresholds, which are designed to detect edges as high intensity peaks, give large thick lines when dealing with SICNN outputs due to their requirement to set a percentage of pixels as edges. Other local thresholds, which tend to use local averages, do not perform very accurately tending to find the edges on each side of a processed image. This leaves edges outlined on both sides but the actual edge is suppressed.

Some study of the output histogram of the SICNN edge detectors, which takes into account the knowledge that the relative changes in output levels give similar sized peaks, would allow a more accurate thresholding mechanism to be developed.

Optimisation of the parameters of the SICNN Complementary Output Processing and

SICNN Division Output Processing edge detectors found that the results of the previous studies on COP optimisation disagreed with the results of this study. Some reasons for this are discussed in chapter 5. The β value for the optimised COP SICNN was 0 and the β value for the optimised DOP SICNN was found to be 1.7.

A comparison of the COP and DOP performance shows that DOP outperforms COP in both quality of edge detection and low processing overhead. The future of SICNN edge detection seems to point towards the DOP as the output processing method.

The final stage in the project was the comparison of the SICNN edge detector with COP to the Sobel and Canny edge detectors. The objective results of this comparison again outlined the inadequate performance of the histogram hysteresis threshold method under low ESNR conditions. At high ESNR levels the SICNN detector performed on par with the Canny detector and both performed better than the Sobel detector.

In the subjective test the SICNN was a clear winner. The Canny detector performed poorly when there was high levels of noise on an image while the Sobel often missed edges. It was noted that the performance of the SICNN edge detector is particularly variable as the length of the weight matrix and the level of the threshold is varied. This can be a good thing as it allows easy customisation of the edge detector to different images.

One of the biggest effects of the SICNN edge detector was the ability of it to suppress areas of similar intensity. This means that rather than having small noisy peaks in the background the areas are almost completely flat. Real edge peaks are of high enough levels that the thresholding scheme would not select any peaks that did occur in the background areas.

The performance of the SICNN edge detectors is very good under many different conditions. The ability of the SICNN to be implemented easily in VLSI, due to its CNN structure, means that they have a future as on chip vision pre-processors.

The development of SICNNs has been based on biological studies of real vision systems. It seems certain that future artificial vision systems will have, as part of their structure, a place for early levels of visual processing. SICNN implementation on chip will perform this function nicely.

8.2 Areas of Further investigation

There are still many areas that could be investigated in SICNNs.

An obvious area would be the implementation of a genetic algorithm to investigate connection weights for the various types of SICNN with output processing. There are a number of Matlab toolboxes that exist in public domain which provide algorithms for mutation between generations and provide guides in writing selection algorithms and defining the genetic characteristics of the connection weight matrix. A useful toolbox for this is the GAOT toolbox; a URL is provided for this in the references.

A useful tool for the study of the parameters available within the SICNNs with output processing would be to implement the SICNNs as a plug-in for Adobe Photoshop or Paintshop Pro. Plug-ins can provide an interactive dialog box which would enable easy modification of the parameters while being able to view the output immediately. A plug-in software development kit exists and is free from the Adobe WWW site. A URL is provided for this in the references.

Tests comparing the DOP SICNN with other edge detectors would also be interesting with the new optimum DOP SICNN showing promise as being more effective than the optimum COP SICNN. This could look more closely at different methods for evaluating two-dimensional edge detector performance.

The analysis of colour images using SICNN is also an area that would be interesting. Possibilities for improving the performance in areas of low contrast may be possible by examining the separate colours. As combining the colours to make a grey scale intensity image suppresses the individual colour so too it could suppress edges that would be visible in the separate colour images. Some form of combination, similar to the hysteresis thresholding, which separates the image into low intensity edges and

high intensity edges and then adds the low intensity edges where they connect to the high intensity edges would be a practical consideration. The edges would be detected in each colour image and in the combined grey scale image. Each edge image could use a high threshold and then the colour images could be combined with the grey scale to improve the performance of the edge detector.

References

Abdou, I.E. & Pratt W.K. (1979). Quantitative Design and Evaluation of Enhancement/Thresholding Edge detectors. Proc. IEEE, Vol. 67-5, pp. 753-763, May.

Adobe Software Development Kit. (2000). [on-line]. Available: http://partners.adobe.com/asn/developer/gapsdk/PhotoshopSDK.html

Barlow, H.B. (1981). The Ferrier Lecture. Critical Limiting Factors in the Design of the Eye and the Visual Cortex. Proc. R. Soc Lond., vol. B212, pp. 1-34.

Bouzerdoum, A. & Pinter, R.B. (1993). Shunting Inhibitory Neural Networks: Derivation and Stability Analysis. IEEE Trans. on Circuits and Systems – I, vol. 40, no.3, pp. 215-221, March.

Brodie, S.E., Knight, B.W., & Ratliff, F. (1978). The Spatiotemporal Function of the Limulus Lateral Eye. J. General Physiology, vol.72, pp. 167-202.

Canny, J. (1986). A Computational Approach to Edge Detection. IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 8, no.6. pp.679-698, November.

Chua, L.O. & Yang, L. (1988). Cellular Neural Networks: Theory. IEEE Trans. Ion Circuits and Systems, vol. 35, no. 10, pp. 1257-1272, October.

Hartline, H.K. & Ratliff, F. (1957). Inhibitory Interactions of Receptor Units in the eye of the Limulus. J. General Physiology, vol. 40, pp. 357-376.

Hartline, H.K. & Ratliff, F. (1958). Spatial Summation of Inhibitory Influences in the Eye of Limulus, and the Mutual Interaction of Receptor Units. J. General Physiology, vol. 41, pp. 1049-1066.

Heath, M., Sarkar, S., Sanocki, T. & Bowyer, K. (1996). Comparison of Edge Detectors: A Methodology and Initial Study. Proc. IEEE Comp. Soc. Conf. On Computer Vision and Pattern Recognition, San Francisco, pp. 143-148

Houck, C.R., Joines, J.A. & Kay M.G. A Genetic Algorithm for Function Optimisation: A Matlab implementation [on-line]. Available: http://www.eos.ncsu.edu/eos/service/ie/research/kay_res/GAToolBox/gaot/

Jernigan, M.E. & McLean, G.F. (1992). Lateral Inhibition and Image Processing, chapter 17, pp. 451-463. Boca Raton: CRC Press.

Mach, E. (1886a). Uder den Physiologischen Effect Raumlich Verteiler Lichtrieze. Sitzungsberichte der mathematisch-naturwissenschaftlichen Classe der kaiserlichen Akademie der Wissenschaften, vol 54 II, no. 134, pp. 131-144.

Mach, E. (1886b). Uder die Physiologischen Wirkung Raumlich Verteiler Lichtrieze III. Sitzungsberichte der mathematisch-naturwissenschaftlichen Classe der kaiserlichen Akademie der Wissenschaften, vol 54 II, no. 134, pp. 393-408.

Mead, C. (1989). Analog VLSI and Neural Systems. Reading, Massachusetts: Addison Wesley Publishing Company.

Nabet, B. & Pinter R.B. (1991). Sensory Neural Networks: Lateral Inhibition. Boca Raton: CRC Press.

Paradis, M.A.K., & Jernigan, M.E. (1994). Homomorphic vs Multiplicative Lateral Inhibition Models for Image Enhancement. IEEE Int. Conf. Syst., Man and Cybernetics, pp. 286-291

Parker, J.R. (1997). Algorithms for image processing and computer vision. New York: John Wiley & Sons, Inc.

Pinter, R.B. (1983a). Product Term Nonlinear Lateral Inhibition Enhances Visual Sensitivity for Small Objects or Edges. J. Theoretical Biology, vol. 100, pp. 525-531

Pinter, R.B. (1983b). The Electrophysiological Bases for Linear and Nonlinear product Term Lateral Inhibition and the Consequences for Wide Field Textured

Stimuli. K. Theoretical Biology, vol. 105, pp.233-243.

Pitas, I. (1993). Digital Image Processing Algorithms. Hertfordshire: Prentice Hall.

Pontecorvo, C. (1998). Edge Detection and Enhancement Using the Shunting Inhibitory Cellular Neural Networks. The University of Adelaide: Adelaide.

Ratliff, F., Hartline, H.K., & Miller, W.H. (1963). Spatial and Temporal Aspects of Retinal Inhibitory Interactions. J. Opt. Soc. Am., vol. 53, pp. 110-120.

Srinivasan, M.V., Laughlin, S.B., & Dubs, A. (1982). Predictive Coding: a Fresh View of Inhibition in the Retina. Proc. R. Soc. Lond., vol. B216, pp. 427-459.

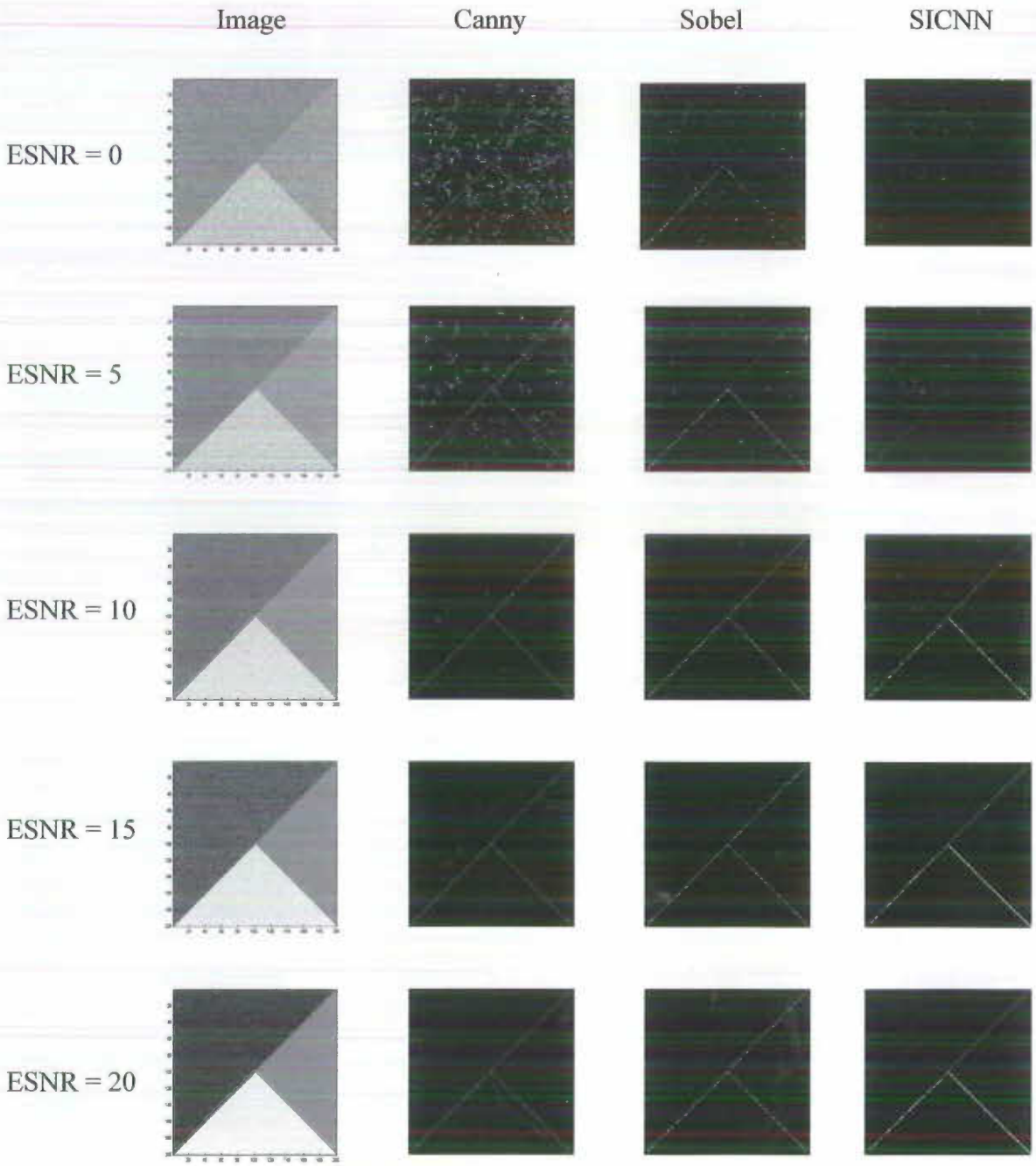
Van der Heyden, F. (1993). Evaluation of Edge Detection Algorithms. 3rd Int. Conf. On Image Processing and its Applications, Warwick, pp. 618-622

Ward, J. (1999). SICNN Optimisation for Edge Detection and Image Enhancement. Edith Cowan University: Perth.

Wasserman, P. D. (1989). Neural Computing Theory and Practice. New York: Van Nostrand Reinhold.

Appendix A

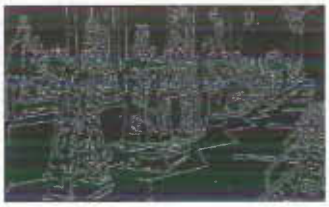
Objective Images - Chapter 7

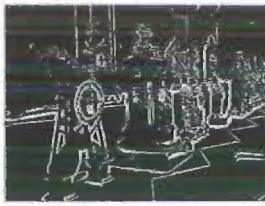


Appendix B

Images and raw results – Chapter 7

Image 1

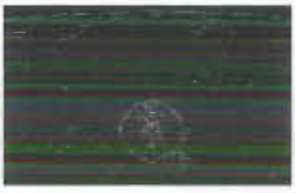
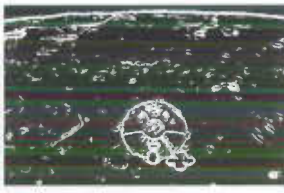





Canny Sobel SICNN

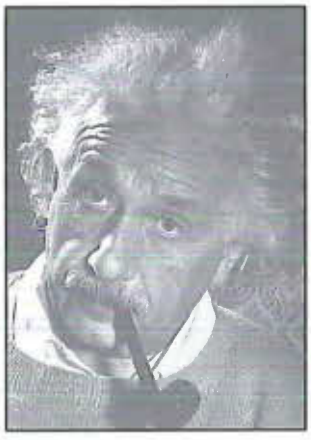
Image 2



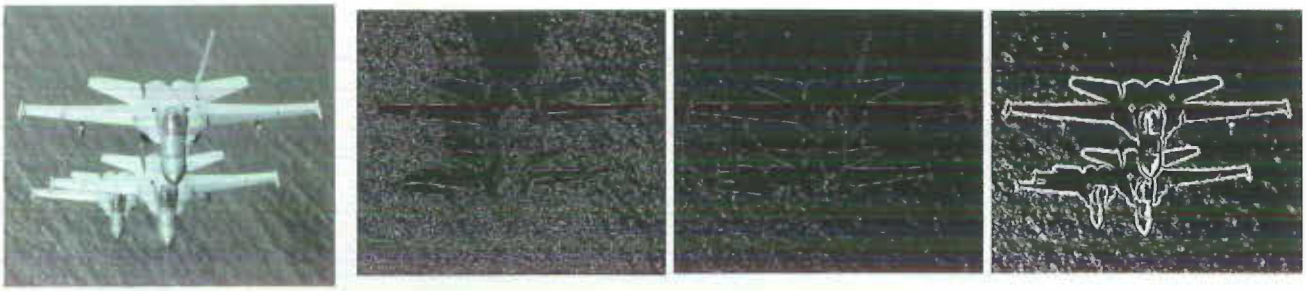
Canny Sobel SICNN

Image 3




Canny Sobel SICNN

Image 4

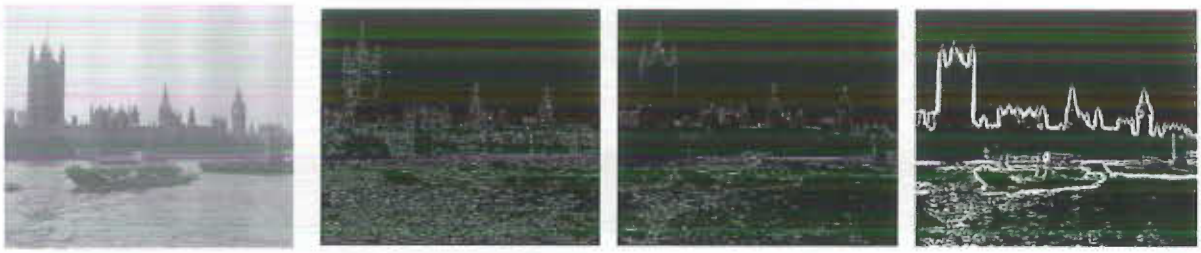


Canny Sobel SICNN

Image 5

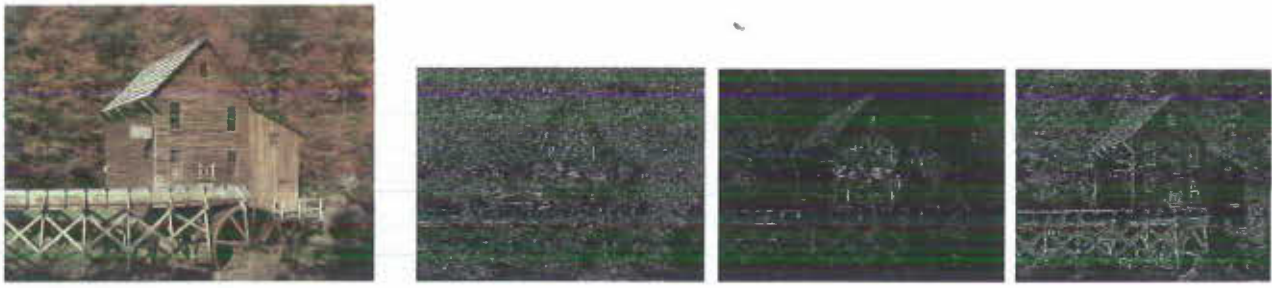
Canny Sobel SICNN

Image 6



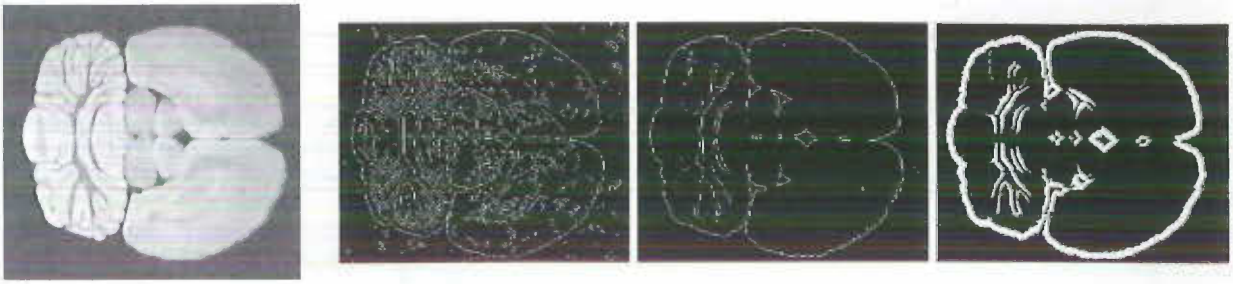
Canny Sobel SICNN

Image 7



Canny Sobel SICNN

Image 8

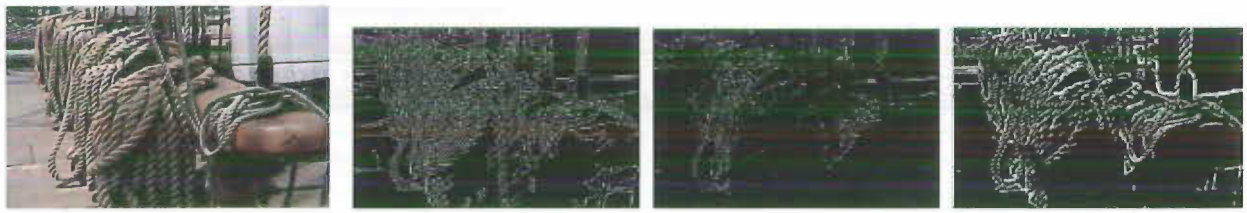


Canny Sobel SICNN

Image 9

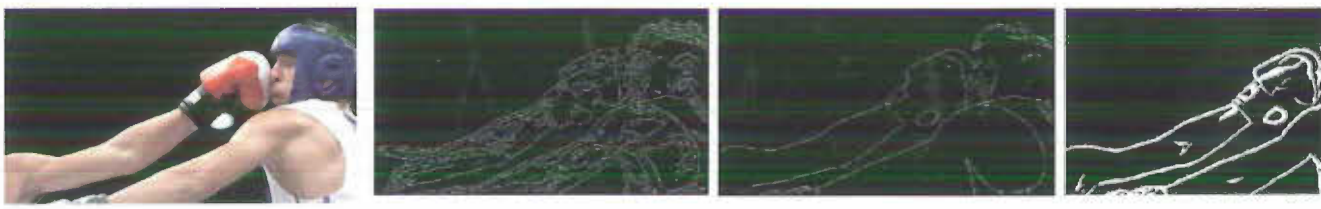
Canny Sobel SICNN

Image 10



Canny Sobel SICNN

Image 11



Canny Sobel SICNN

The following table contains the raw scores for the subjective image test.

	Subjects									
Image1	1	2	3	4	5	6	7	8	9	10
Sobel	1	1	1	1	1	1	1	2	1	1
Canny	2	3	3	3	3	3	3	3	2	3
SICNN	3	2	2	2	2	2	2	1	3	2
Image2	1	2	3	4	5	6	7	8	9	10
Sobel	2	3	2	1	1	3	3	3	1	3
Canny	1	1	3	3	3	1	1	1	2	1
SICNN	3	2	1	2	2	2	2	2	3	2
Image3	1	2	3	4	5	6	7	8	9	10
Sobel	2	1	1	1	1	1	1	2	1	1
Canny	1	2	2	3	3	2	2	1	2	2
SICNN	3	3	3	2	2	3	3	3	3	3
Image4	1	2	3	4	5	6	7	8	9	10
Sobel	2	3	2	1	1	2	2	2	1	2
Canny	1	1	3	3	3	1	1	1	2	1
SICNN	3	2	1	2	2	3	3	3	3	3
Image5	1	2	3	4	5	6	7	8	9	10
Sobel	3	3	2	2	1	1	3	1	1	3
Canny	1	2	1	3	3	2	2	3	3	1
SICNN	2	1	3	1	2	3	1	2	2	2
Image6	1	2	3	4	5	6	7	8	9	10
Sobel	3	2	3	2	2	1	2	3	2	3
Canny	1	3	2	3	3	2	3	1	3	2
SICNN	2	1	1	1	1	3	1	2	1	1
Image7	1	2	3	4	5	6	7	8	9	10
Sobel	2	2	1	1	1	2	1	1	2	1
Canny	1	1	3	3	3	1	2	2	1	2
SICNN	3	3	2	2	2	3	3	3	3	3
Image8	1	2	3	4	5	6	7	8	9	10
Sobel	2	3	2	2	3	1	1	3	3	2
Canny	1	2	3	3	1	3	3	1	2	3
SICNN	3	1	1	1	2	2	2	2	1	1
Image9	1	2	3	4	5	6	7	8	9	10
Sobel	2	1	2	2	1	1	2	2	1	1
Canny	3	3	1	1	3	2	3	1	2	2
SICNN	1	2	3	3	2	3	1	3	3	3
Image10	1	2	3	4	5	6	7	8	9	10
Sobel	1	1	1	1	1	1	1	1	1	2
Canny	2	2	3	2	3	2	3	2	2	1
SICNN	3	3	2	3	2	3	2	3	3	3
Image11	1	2	3	4	5	6	7	8	9	10
Sobel	1	1	1	1	1	1	1	2	1	1
Canny	2	3	3	2	3	2	2	1	2	2
SICNN	3	2	2	3	2	3	3	3	3	3

Appendix C

SICNN toolbox structure

There are many files contained in the SICNN toolbox structure. Following are a number of files that are included to do many of the functions required.

C.1 Contents.m

```
% SICNN Toolbox.
% This is written to demonstrate the shunting inhibitory cellular neural network
%   edge detectors.
%
%Processing:
% SICNN2d - this implements the two-dimensional sicnn edge detector.
% sicnn1 - one dimensional edge detector.
% sicnnCOP - complementary output processing with SICNN.
% sicnnDOP - division output processing with SICNN.
%
%Thresholding:
% rawthresh - raw threshold on a threshold level.
% movavthresh - moving average threshold.
% twopeakthresh - 2 peaks threshold.
% hyshistthresh - hysteresis histogram threshold (as in edge.m).
%
%   Grant Walker, 3 November 2000.
%   Copyright (c) 2000 by G. Walker.
%   All rights reserved.
```

C.2 Hyshistthresh.m

```
function Eout=hyshistthresh(I,percentpixels);
%hyshistthresh - hysteresis histogram method
%       as used in edge.m
%
%  Eout = hyshistthresh(I,percentpixels)
%  INPUT:
%       I    = input image.
%       percentpixels = percentage of image which is edge pixels.
%
%  OUTPUT:
%       Eout = output edge image.
%
%   Adapted from edge.m Grant Walker, 3 November 2000.
%   Copyright (c) 2000 by G. Walker.
%   All rights reserved.

[p,q]=size(I);
PercentOfPixelsNotEdges=percentpixels;
ThresholdRatio=0.7;
[counts,x]=imhist(I, 64);
highThresh = min(find(cumsum(counts) > PercentOfPixelsNotEdges*p*q)) / 64;
lowThresh = ThresholdRatio*highThresh;
```

```
thresh = [lowThresh highThresh];

Eweak = I >lowThresh;
Estrong=I>highThresh;


[rstrong, cstrong]=find(Estrong);
e = bwselect(Eweak, cstrong, rstrong, 8);
Eout=e;
```

C.3 Movavthresh.m

```
function Eout=movavthresh(I)
%movavthresh - moving average threshold
%
%  Eout = movavthresh(I)
%  INPUT:
%       I    = input image.
%
%  OUTPUT:
%       Eout = output edge image.
%
%   Grant Walker, 3 November 2000.
%   Copyright (c) 2000 by G. Walker.
%   All rights reserved.
[NR NC]=size(I);
%used to create initial average
s=8;
sum=0;
for i=1:NR
   if rem(i,2)%odd row
      for j=1:NC
         sum=sum-sum/s+I(i,j);
         av=sum/s;
         if I(i,j)>2*av
            Eout(i,j)=255;
         else
            Eout(i,j)=0;
         end
      end
   else
      for j=NC:-1:1
         sum=sum-sum/s+I(i,j);
         av=sum/s;
         if I(i,j)>2*av
            Eout(i,j)=255;
         else
            Eout(i,j)=0;
         end
      end
   end
end
```

C.4 Rawthresh.m

```
function Eout=rawthresh(I,v)
% rawthresh - raw threshold using value v
%
```

```
%  Eout = rawthresh(I,v)
%  INPUT:
%        I    = input image.
%        v    = threshold value.
%
%  OUTPUT:
%        Eout = output edge image.
%
%   Grant Walker, 3 November 2000.
%   Copyright (c) 2000 by G. Walker.
%   All rights reserved.

Eout = (I > v);
Eout=I2*255;
```

C.5 Sicnn1.m

```
function I = sicnn1(X, C, a, s);
%
%sicnn1 - Implements Shunting Inhibitory Cellular Neural Networks.
%        [I, L] = sicnn(X, N, s, n) finds 2 outputs images I and L.
%
%  INPUT:
%        X   - Input Image.
%        N   - Number of Iterations.
%
%  OUTPUT:
%        I   - First output Image.
%        L   - Second output Image.
%
%   Abdesselam Bouzerdoum, 28 December 1993.
%   Copyright (c) 1993 by A. Bouzerdoun.
%   All rights reserved.
%
%      modified 9 September 1999
%  modified 29 September 2000 - Grant Walker
%   1. included Io in the decay factor of sicnn mathematics.
%   2. modified to work with any sized C matrix

if nargin == 1
     a = 1;
     C = [0 0 0 0 boxcar(3)'];
     s  = 0;
  elseif nargin == 2
     a  = 1;
     s  = 0;
  elseif nargin == 3
     s  = 0;
  else
  end

C = C/sum(C);

%append columns for horizontal sicnn
n = fix((size(C,2)-1)/2);
for i = 1:2*n
   [p,q]= size(X);
    X = [X(:,1), X, X(:,q)];
end

%append rows for vertical sicnn
m = fix((size(C,1)-1)/2);
for i = 1:2*m
```

```
    [p,q]= size(X);
    X = [X(1,:); X; X(p,:)];
end

%smoothing algorithm
if s ~= 0
    [x,y] = meshgrid(-n:n,-n:n);
       g = exp(-(x.^2+y.^2)/(2*s^2));
       g0= sum(sum(g));
       X = conv2(X,g,'same')/g0;
end


%create mean matrix(Io) for decay rate
%[p,q]=size(C);
%MF=[(1/(p*q))*ones(p,q)];
%Io=conv2(X,MF,'same');


   [P,Q] = size(X);
   I = X./(a + conv2(X,C,'same'));
%  I = X./(a*Io + conv2(X,C,'same'));
   I = I(:,2*n+1:Q-2*n);
   I = I(2*m+1:P-2*m,:);
   %  I = I(2*m+1:P-2*m,2*n+1:Q-2*n);
```

C.6 Sicnn2d.m

```
function [eout]=sicnn2d(I,Method,Threshold,C,alpha)
%SICNN2d tool
%
%  This program takes in the image and processes it according to one of three
%  outputprocessing methods.  A thresholding stage in included.
%
%        Eout = sicnn2d(I,Method,Threshold,C,alpha) finds output images Iout.
%          or
%        sicnn2d(I,Method,Threshold,C,alpha) which displays image using
imshow(Iout)
%  INPUT:
%        I    - Input Image.
%        Method    - 's' standard sicnn, 'cop' complementary output processing,
%                    'dop' division output processing.
%        Threshold    - 'ma' moving average, '2p' two peaks, 'edge' hysteresis
histogram
%                         (as used in edge.m), 'none' no thresholding.
%        C    - Connection weight matrix.
%        alpha    - decay rate.
%
%  OUTPUT:
%        Eout    - Edge output image
%
%  Defaults: Method = 'dop', Threshold = 'none', C=optimum, alpha=2*(average
intesity)
%
%  Each of the thresholds is also available to access individually
%      ma = movavthresh.m
%      2p = twopeakthresh.m
%      edge = hyshistthresh.m
%
%   Grant Walker, 3 November 2000.
%   Copyright (c) 2000 by G. Walker.
%   All rights reserved.
```

```
error(nargchk(1,6,nargin));


methods={'s','cop','dop'};
thresholds={'2p','none','edge','ma'};

if isrgb(I),
   error('RGB images not supported, call RGB2GRAY');
end

if isa(I,'uint8')|isa(I,'uint16'),
   I=im2double(I);
end
%get arguements
if nargin == 1,
     Method= methods{3};
     Threshold = thresholds{2};
     C=[];
     alpha=2*mean(mean(I));
  elseif nargin == 2,
     Threshold = thresholds{2};
     C=[];
     alpha=2*mean(mean(I));
  elseif nargin == 3,
     C=[];
     alpha=2*mean(mean(I));
  elseif nargin == 4,
     alpha=2*mean(mean(I));
  else
  end

  %check method valid
  str=lower(Method);
  J=strmatch(str,methods);
  if isempty(J),
     error(['Invalid SICNN method: 'Method ]);
  end

  %check threshold valid
  str=lower(Threshold);
  J=strmatch(str,thresholds);
  if isempty(J),
     error(['Invalid SICNN threshold: 'Threshold ]);
  end

  switch Method
  case 's'
     if isempty(C)
        C=[0 0 0 0 1 1 1];
     end
     out=sicnn1(I,C,alpha);
     outmax=max(out(:));
     if outmax>0
        out=out/outmax; %normalise
     end
  case 'cop'
     if isempty(C)
        C=[0 0 0 0 1 1 1];
     end
     out=sicnnCOP(I,C,alpha);
  case 'dop'
     C=kaiser(8,1.7);
     C=rot90([-1*C(1:4); C(5:8)]);
     out=sicnnDOP(I,C,alpha);
  otherwise
  end
  switch Threshold
  case 'edge'
```

```
    out=hyshistthresh(out,0.9);
  case 'ma'
    out=movavthresh(out);
  case '2p'
    out=twopeakthresh(out);
  case 'none'
  otherwise
  end

  if nargout==0,
    imshow(out);
  else
    eout=out;
  end
```

C.7 Sicnncop.m

```
function Eout=sicnnCOP(I,C,alpha)
%sicnnCOP - Processes image using SICNN with complementary output processing
%      Eout=sicnnCOP(I,C,alpha)
%
%  INPUT:
%       I    = input image.
%       C    = connection weights matrix
%       alpha = decay rate
%
%  OUTPUT:
%       Eout = output edge image.
%
%   Grant Walker, 3 November 2000.
%   Copyright (c) 2000 by G. Walker.
%   All rights reserved.
C = C/sum(C);

C1=C;
C2=fliplr(C);
C3=rot90(C);
C4=rot90(C,-1);

%Sicnn processing
Ixr = sicnn1(I,C1,alpha);
Ixl = sicnn1(I,C2,alpha);
Iyr = sicnn1(I,C3,alpha);
Iyl = sicnn1(I,C4,alpha);

%cop
Ix=Ixr-Ixl;
Iy=Iyr-Iyl;

mag=sqrt(Ix.*Ix+Iy.*Iy);


magmax=max(mag(:));
if magmax>0
   mag=mag/magmax; %normalise
end

Eout=mag;
```

C.8 Sicnndop.m

```
function Eout = sicnnDOP(I, C, alpha);
%sicnnDOP - Processes image using SICNN with division output processing
%       Eout=sicnnDOP(I,C,alpha)
%
%  INPUT:
%       I    = input image.
%       C    = connection weights matrix
%       alpha = decay rate
%
%  OUTPUT:
%       Eout = output edge image.
%
%   Grant Walker, 3 November 2000.
%   Copyright (c) 2000 by G. Walker.
%   All rights reserved.


[N1,N]=size(C);

C1=[(1/N)*ones(1,N)];

I1=I;

%append columns for horizontal sicnn
n = fix((size(C,2)-1)/2);
for i = 1:2*n
   [p,q]= size(I1);
    I1 = [I1(:,1), I1, I1(:,q)];
end

[P,Q] = size(I1);
Ix = conv2(I1,C,'same')./(alpha + conv2(I1,C1,'same'));
Ix = Ix(:,2*n+1:Q-2*n);

%append columns for vertical sicnn
for i = 1:2*n
   [p,q]= size(I);
    I = [I(1,:); I; I(p,:)];
end

C=rot90(C);
C1=rot90(C1);
[P,Q] = size(I);
Iy = conv2(I,C,'same')./(alpha + conv2(I,C1,'same'));
Iy = Iy(2*n+1:P-2*n,:);



mag=sqrt(Ix.*Ix+Iy.*Iy);
magmax=max(mag(:));
if magmax>0
   mag=mag/magmax; %normalise
end

Eout=mag;
```

C.9 TwoPeakThresh.m

```
function Eout=twopeakthresh(I)
%twopeakthresh - 2peaks thresholding method
%
%  Eout = twopeakthresh(I)
```

```
%  INPUT:
%        I    = input image.
%
%  OUTPUT:
%        Eout = output edge image.
%
%   Grant Walker, 3 November 2000.
%   Copyright (c) 2000 by G. Walker.
%   All rights reserved.

[counts,x]=imhist(I, 64);
[v,firstpeak]=max(counts);
for i=1:firstpeak
   counts(i)=0;
end
for i=firstpeak+1:64
       counts(i)=counts(i)*(i-firstpeak);
end

[v,secondpeak]=max(counts);
i=secondpeak;

I = (I > (x(i)/64)*max(I(:)));
Eout=I;
```